Spring 5-2011

Cloud Shadow Detection and Removal from Aerial Photo Mosaics Using Light Detection and Ranging (LIDAR) Reflectance Images

Glover Eugene George
University of Southern Mississippi

The University of Southern Mississippi

CLOUD SHADOW DETECTION AND REMOVAL FROM AERIAL PHOTO MOSAICS USING LIGHT DETECTION AND RANGING (LIDAR) REFLECTANCE IMAGES

by

Glover Eugene George

Abstract of a Dissertation
Submitted to the Graduate School
of The University of Southern Mississippi
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

May 2011

ABSTRACT

CLOUD SHADOW DETECTION AND REMOVAL FROM
AERIAL PHOTO MOSAICS USING LIGHT DETECTION AND RANGING (LIDAR)
REFLECTANCE IMAGES

by Glover Eugene George

May 2011

The process of creating aerial photo mosaics can be severely affected by clouds and the shadows they create. In the CZMIL project discussed in this work, the aerial survey aircraft flies below the clouds, but the shadows cast from clouds above the aircraft cause the resultant mosaic image to have sub-optimal results. Large intensity variations, caused both from the cloud shadow within a single image and the juxtaposition of areas of cloud shadow and no cloud shadow during the image stitching process, create an image that may not be as useful to the concerned research scientist. Ideally, we would like to be able to detect such distortions and correct for them, effectively removing the effects of the cloud shadow from the mosaic.

In this work, we present a method for identifying areas of cloud shadow within the image mosaic process, using supervised classification methods, and subsequently correcting these areas via several image matching and color correction techniques. Although the available data contained many extreme circumstances, we show that, in general, our decision to use LIDAR reflectance images to correctly classify cloud and not cloud pixels has been very successful, and is the fundamental basis for any color correction used to remove the cloud shadows. We also implement and discuss several color transformation methods which are used to correct the cloud shadow covered pixels, with the goal of producing a mosaic image which is free from cloud shadow effects.

The University of Southern Mississippi

CLOUD SHADOW DETECTION AND REMOVAL FROM AERIAL PHOTO MOSAICS USING LIGHT DETECTION AND RANGING (LIDAR) REFLECTANCE IMAGES

by

Glover Eugene George

A Dissertation
Submitted to the Graduate School
of The University of Southern Mississippi
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

Approved:

Benjamin Seyfarth
Director

Dia Ali

Beddhu Murali

Ras Pandey

Chaoyang Zhang

Susan A. Siltanen
Dean of the Graduate School

May 2011

ACKNOWLEDGMENTS

This is to thank all of those who have assisted me in this effort. I am forever indebted to my advisor, Dr. Benjamin Seyfarth, who is the source of all wisdom in this worldly life. I am and remain in his awesome, brilliant shadow.

Thanks also to the School of Computing at Southern Miss, that so graciously allowed me to slack on system administration duties at times, in order to complete this research.

Also, special thanks goes to Optech, Inc. for providing the data on which this research was based. Obviously, none of this would have been possible without their help.

Lastly, thanks to family and friends for their patience and understanding during this process, especially the ones who have been patiently waiting on their fishing trips. I might have some free time now.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Figure

LIST OF ABBREVIATIONS

SVM	-	Support Vector Machine
LIDAR	-	Light Detection and Ranging
CCM	-	Consistent Color Mapping
RGB	-	Red, Green, and Blue
CZMIL	-	Coastal Zone Mapping and Imaging Lidar
JALBTCX	-	Joint Airborne Lidar Bathymetry Technical Center of Expertise
HSI	-	Hue, Saturation, and Intensity
HSV	-	Hue, Saturation, and Value
DHW	-	Dynamic Histogram Warping
LIBLINEAR	-	Linear Classification Library
LIBSVM	-	Support Vector Machine Library
MMH	-	Maximal-Margin Separating Hyper-Plane
PGPDT	-	Parallel Gradient Projection-based Decomposition Technique
C-SVM	-	Support Vector Classification
ν**-SVM**	-	Support Vector Regression
RBF	-	Radial Basis Function
nm	-	Nanometers
avg	-	Average

Chapter 1

BACKGROUND

1.1 Introduction

Aerial photo-mosaics are becoming more prevalent with advances in aerial photography. Many new types of aircraft, such as unmanned aerial vehicles, rely heavily on detailed satellite and aerial surveillance photography to navigate, perform change detection in an area, and deploy weapons systems against ground-based targets. Civilian and commercial applications are also being developed which are built upon these image mosaics, such as Google Earth and Tele-Atlas. City planning, Global Positioning System (GPS)-based navigation systems and news organizations all make extended use of overhead, aerial photo-mosaics. While many of these mosaics primarily utilize satellite-based images, for many applications, including The Coastal Zone Mapping and Imaging LIDAR (CZMIL) project from which this research was born, only low-flying aircraft equipped with very high resolution cameras can provide the type of detailed data required for specific applications.

The CZMIL project is a joint effort by the U.S. Army Corps of Engineers and the Joint Airborne LIDAR Bathymetry Technical Center of Expertise (JALBTCX) to develop software and hardware systems to map the coastlines of waterfront land areas, including both the nearby land masses and the bottom surface of the shallow coastal waters. Using light detection and ranging equipment (LIDAR), the aircraft is able to constantly measure the height of the ground below it, including the bottom depth of the nearby coastal waters (typically 1km offshore). Simultaneously, the aircraft acquires high resolution photographs of the surface below, and once the mission is finished, the photographs are downloaded and processed off-line. The thousands of RGB images are then stitched together to form one

large photo-mosaic. Researchers can then use these mosaics to better understand the other types of data collected on the flight, such as hyper-spectral, infrared, and bathymetry data.

In the CZMIL project, the objective is to map the ocean floor along coastal areas, as well as the surrounding land, in order to build maps and models of the underlying bathymetry. This is important in areas where traditional mapping techniques are insufficient, such as shallow coastal zones where large survey ships cannot go due to draft and depth issues, or to avoid distortions caused from their propellers disturbing the sandy bottoms. Certain frequency ranges of LIDAR can penetrate the water, and their measured reflectance is used to measure points of depth on the bottom surface. This elevation (depth) data can then be used to build a 3D model, which is used both in the geo-referencing of the image data, as well as building visualizations and maps.

The LIDAR system in use in the CZMIL project is an Optech SHOALS-1000T integrated sensor array. This package includes a 1,000 pulse per second bathymetric laser used for penetrating bodies of water, which operates at around 532 nm. A second laser operating at 9,000 pulses per second (at 1064nm) is used for measuring topographic reflections over both land and water. The difference in reflection times between the bathymetric laser and the topographic laser (which reflects at the surface of the water) is what is used to calculate the water depth at specific points.

Hyper-spectral data is also collected via a CASI-1500 imager, operating in a forward, push-broom fashion. Researchers can select up to 288 separate bands of data, configurable from 375 nm to 1050nm. Depending on the number of bands in use and the type of data being collected, ground pixel resolutions are capable between 20cm and 5m per pixel. The final piece of equipment in the SHOALS package is the RGB camera, which produces images with a ground resolution of up to 20 cm per pixel. The methods described here, however, are limited to the two lasers and the RGB camera, as hyper-spectral data was not available.

1.2 Problems with the Photo Mosaic Process

In the CZMIL project, the survey aircraft flies at a relatively low altitude and always flies beneath the clouds. This prevents the clouds themselves from interfering with the data readings from the LIDAR lasers, the hyper-spectral imager, and the RGB photographs. However, this does not prevent shadows cast from clouds located above and around the aircraft from obscuring large portions of the earth being examined. The sun is, obviously, the primary illuminant when capturing overhead images of the earth, and it is this light reflection from the surface of the earth to the camera lens which forms our image. Clouds absorb different wavelengths of light in different ways, but in general the overall effect is that it reduces the intensity of light which is then reflected back to the camera lens. This results in darker images, as you can see in Figure 1.1 below, which may lack enough detail and information to be useful to the researcher.

Yet another problem that cloud shadows cause in image mosaics arises when we consider how the images are acquired. The aircraft uses a flight grid in which it flies one direction (maybe due east) and acquires images in a straight line. The aircraft maintains a constant speed which is selected to correspond to the data acquisition frequency requirements of the survey mission (such as distance between samples, resolution, etc). This helps to ensure that there will be adequate overlap between each successive image in an effort to cover the entire area below. Once the initial path is complete, the aircraft circles around and begins acquiring data in the reverse direction (west for this example) parallel to the original path, and attempts to overlap data acquisition in both its own flight path direction and the previous easterly flight path. This also help prevent gaps between successive flight lines.

One obvious problem with this scenario is the position of the sun relative to the direction of the flight path, which changes constantly during the day in the east-west directions, and north-south throughout the different seasons. While the flights are generally completed during a single day, even in the best case scenario they still cover many hours on either side

Figure 1.1: An image which is partially occluded by cloud shadow.

of Noon. When the images are being acquired during the morning hours, easterly flight lines will tend to produce images with higher reflectivity of the sun's light, as the majority of the sun's light is reflected from the surface of the earth, directly into the camera (Figure 1.2).

Figure 1.2: Mosaic created using only westerly flight lines. Notice the glinting on the water which is not present in Figure 1.3.

Figure 1.3: Mosaic created using only easterly flight lines.

However, the westerly flight lines (Figure 1.3) will not receive the same amount of light reflection into the aperture of the camera, and thus these images will be less intense (less bright) and will not be as adversely affected by the differences in surface material reflectance properties, such as that of water and concrete. In Figure 1.4, one can clearly see the effects of the low angle of the sun on a parking lot and the tops of several buildings. Although the surfaces are not truly white (with RGB intensities of 255, 255, and 255, respectively), the high reflectivity of the material causes the resultant image to represent them this way.

Figure 1.4: Mosaic created using only westerly flight lines.

Clouds also generally do not remain stationary during the course of a survey flight mission, further playing havoc with the mosaic creation process. Because the cloud may have been present during an easterly flight line does not necessarily mean it will still be in the same location, casting its shadow on the same area of the ground when the aircraft

comes back on its westerly flight line. In some cases, the cloud may even move a significant amount during the same flight line, between groups of successive images. This along with the previously mentioned problems results in a mosaic which has many visible seams between flight lines, lines between sequential images, and wide intensity variations throughout the image. The dataset provided for this research is, admittedly, an extreme case, as can be seen in Figure 1.5. Note the random cloud shadow locations, the flight paths, and the overall intensity variations.

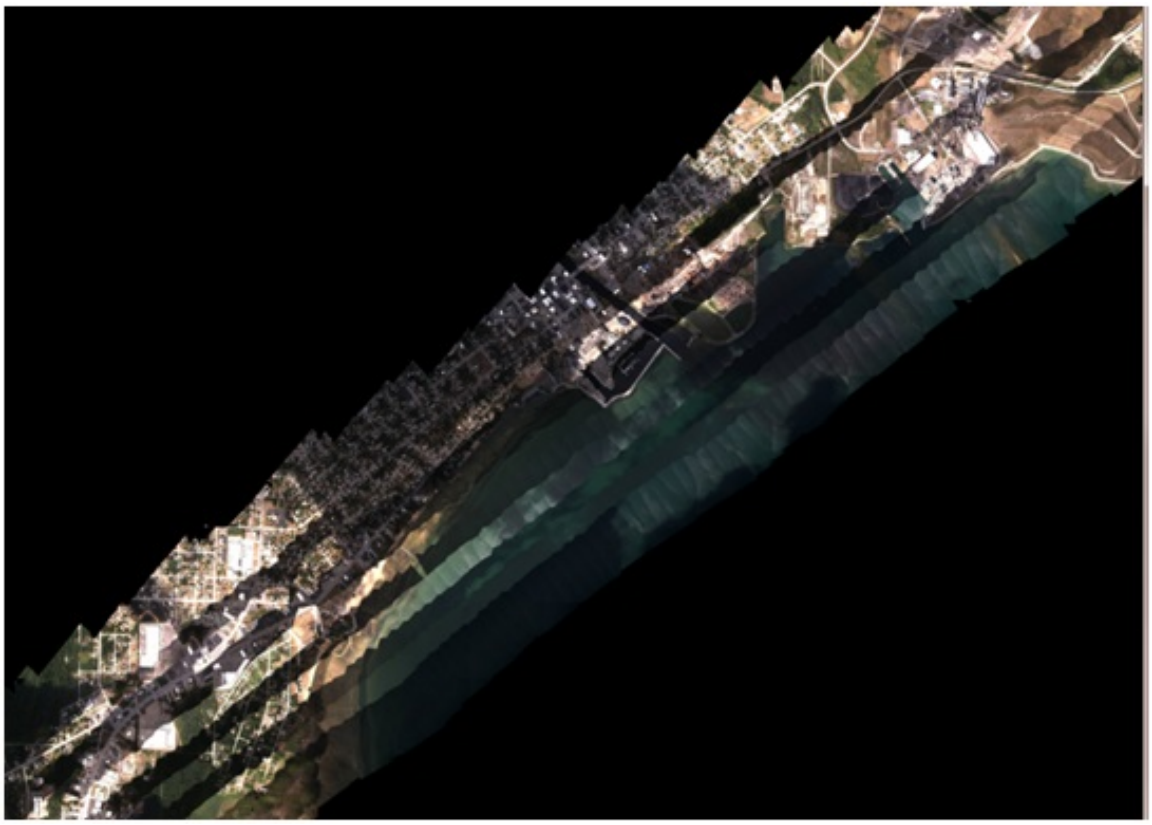

Figure 1.5: Mosaic created using only westerly flight lines.

The objective of this research is to find a way to detect and remove these cloud shadows and intensity variations, and instead produce an image mosaic which appears more uniform in intensity and more closely represents the way the surface being surveyed would have looked had there been a clear, un-obscured, cloud shadow-free view of the surface. While there have been many attempts at cloud and cloud shadow removal in satellite images and

mosaics, there are unique problems and solution possibilities which make ours a relatively novel approach.

1.3 Previous Research

There have been many previous attempts to remove clouds and cloud shadows from satellite and aerial imagery. The majority of the cloud shadow removal techniques from satellite imagery depend on having multi-temporal data for a given area. Various methods are used to determine a good, cloud-free image to use in place of the given image. In [9], Helmer and Reufenacht produce a strategy of cloud-free image mosaics using regression trees and histogram matching. They begin by using regression trees to predict image data underneath clouds and cloud shadows from other scene dates, and then use histogram matching to color match the composited piece to the surrounding, overlapping images. Again, what we would like to do is enhance the data under the cloud shadow from images taken within a relatively short period of time, and therefore do not have access to many scene dates. In [25] the authors develop a method to identify and remove clouds and cloud shadows by using an image fusion technique which integrates complementary information into the composite image from multi-temporal images. The area under the cloud has its brightness changes smoothed due to the cloud shadow, and wavelet transforms are used to identify these regions.

In [4], Dare describes an algorithm for the detection and removal of ground feature shadows from urban imagery. While the author does not concern himself with cloud shadow removal, he does present a thresholding method for detection of building shadows. Dare shows that a threshold can be determined by taking the mean of the bimodal peaks of the histogram of the panchromatic image. In [24] a compositing technique is developed which models the radiance of a pixel as five components, including path radiance and direct radiation. The authors simulate the reflectance values of non-shadowed and shadowed pixels using geometric simulation code requiring inputs such as the solar angle, satellite angle, relative azimuth, etc.

Yet another research area is in how to identify and remove shadows based on the geometric model of where the cloud is located, its height, and the location of the sun, and projecting where the shadow should be [23]. This is then used to try to identify an area as cloud-covered and either try to correct the projected area or match morphologically the shape of the cloud and the shape of the cast shadow. Unfortunately, we have no clouds in our images, and therefore cannot construct any geometric models by which to determine the cloud cover. Even if we did, we are not taking an image of the entire cloud and cloud shadow (due to the low altitude and multiple passes over a short period of time). Once we reconstruct the image via the mosaic, we will not have consistent, unmodified cloud shadow shapes. The clouds will have moved, changed shape, and we may not have the original cast-shadow shape anymore.

An important distinction to make in regards to the majority of satellite-based imagery is that the area imaged by the camera is not only very large, it is also taken from very long distances. For the most part, at this distance any cloud cover will completely obscure the data under the cloud. Due to the dispersion of the reflected light through the atmosphere, very little information about what is actually under the cloud reaches the sensor array. Therefore, any method to reconstruct the underlying data must be either based on images of that region from another cloud-free time period or must be inferred from the remaining surrounding areas.

In our case, the images are taken from low flying aircraft below the clouds, and rarely would there be any direct occlusion of the underlying data by the clouds. The only information loss we have is a diminished intensity due to the cloud casting a shadow over the area and the differences in viewpoint in relation to the illuminant (sun). Since we still have the information about the surface, we simply need to transform the pixel intensity distributions to those of an image which is nearby and not undergoing cloud-shadow effects.

Another major area of interesting related work is in image color matching. The majority of these methods are broken down into one of two categories; those which rely on a pixel-to-

pixel correspondence between a reference and target image [24],[20],[15], and those which rely only on the statistical distribution of pixel values [22],[14],[13],[25],[26],[27]. In [15] Kagarlitsky proposes a method for piece-wise color mappings of images acquired under various conditions. As the author is concerned with applying a precise change detection method, he cannot simply rely on human perception of the transformation quality but rather would like to guarantee a consistent color mapping (CCM). For a CCM to exist, two identical colors in the source image must remain identical to each other after applying a mapping function. They must also remain monotonically increasing; i.e. if color $a_i < a_j$ in the source image, then this relationship must remain so after the mapping.

In order to test for consistency, the author shows that the joint histogram of the two images will show a monotonically increasing line, as opposed to the more random pattern which exists when there is no CCM. However, due to illumination or viewpoint differences a global CCM for the two images may not exist. In this case, Kagarlitsky uses a planar co-segmentation of sub-regions over the overlapping parts of the images and computes individual CCM's. The choice of sub-regions is limited to planar regions as the variations in the intensities of corresponding pixels depend on their surface normal, and those pixels on the same plane will have the same normal. However, the author does not concern himself with cloud cover, and mappings between multiple overlapping segments may vary widely and therefore may not be suitable building blocks for a global CCM. Also, it remains to be seen as to how well a planar co-segmentation would work for aerial or satellite imagery due to the imaging range.

The remaining research uses some form of statistical distribution transformation to map the color from one image to another. It is from these methods that we hope to find promising solutions.

Reinhard [21] describes a method for a general form of color correction between images, in which the goal is to apply the colors from one image to those of another. The author uses $L\alpha\beta$ color space conversions to minimize effects of the tight correlation between bands

in RGB color space. Once in $L\alpha\beta$, the mean and standard deviation are used to scale the data of the reference image to that of the new, synthesized image. Wang et. al. [18] extend Reinhard's work to image sequences and Huang [13] further extend this work to synthesize a new image sequence by using colors from multiple reference images under user specified weights. Also, Zheng et. al. [29] compare the RGB and HSI color spaces for color matching and conclude that the RGB color space is more suited to rectification than HSI. It remains to be seen how $L\alpha\beta$ compares to both RGB and HSI.

Ng and Chu [19] present a color model which is invariant to lighting geometry, illumination color, specularity, and diffuse lighting. As with other image matching models [15], the primary purpose of the author's work is to improve image matching performance, and, as such, attempts are made to model radiometric properties accurately. However, the author does not discuss the applicability of the model when the viewpoint varies, and experimental results in the paper were obtained from datasets with pixel-to-pixel correspondence, and therefore might not be relevant to our current issue.

Roland et. al. [22] discuss histogram matching as it relates to texture synthesis. The authors propose new matching algorithm based on sorting, and show that in small images it outperforms histogram matching. However, the computational complexity for images larger than 64x64 will outweigh the overhead incurred via histogram matching when implemented with indexed look-up tables. Yet when an exact histogram matching is required, the sort-based algorithm proposed would be the more correct choice.

Cox and Hingorani [3] propose yet another histogram matching algorithm in which dynamic programming is used to perform a global optimization that achieves better matching than with straightforward histogram matching alone. In simple histogram matching, values in one bin of the reference image cannot be mapped to more than one value in the target image, leading to reduced precision in the mapping. Dynamic histogram warping (DHW) allows histogram bins to be matched one-to-one, one-to-many (expansion), and many-to one (contraction). The result is an algorithm which can model the relationship between a

pair of images much more accurately than simple additive/multiplicative models.

Jia et. al. [14] compare three histogram-based image matching methods as they relate to vehicle number plate recognition.

Histogram matching, or histogram specification, works by finding a transform that allows us to make one image take on the color distribution of another image. While this method can fail if images are drastically different, such as between an image with mostly land and an image with mostly water, it should work well for our purposes as long as the images in question are of similar, nearby areas. Histogram matching seeks to specify the shape of the histogram we wish the target image to have after transformation. In order to transform an image which has been shadowed to an image without shadow, we first must find the histogram of the reference image and use this as the probability distribution function (pdf) that we use in the following equations.

In Chapter 2, we take a closer look at the LIDAR reflectance image and how it can be used to detect and correct cloud shadow-covered regions of the mosaic. We discuss classification methods and their applicability to the problem of finding cloud shadow in Chapter 3. Color correction methods are explored in Chapter 4, and a more advanced method of mapping the colors from the cloud shadow-covered areas to cloud shadow-free pixels in Chapter 5.

Chapter 2

LIDAR

2.1 LIDAR Reflectance Image

As mentioned earlier, LIDAR is used to provide elevation (and depth) readings from the aircraft to assist in many areas of the research being conducted during the CZMIL flights. When this LIDAR laser is bounced off the surface of the earth, the time difference of the reflected laser light is recorded, and, using the speed of light, the distance can be calculated. One additional feature of the LIDAR which is recorded is the intensity of the laser return. A monochrome image can then be built using these returned intensities and their geo-referenced coordinates, which results in Figure 2.1.

Because the laser is actually using a small portion of visible green light, it can give us a clue as to the green reflectance properties of the surface. As this laser light is an active source of light below the clouds, we obtain this reflectance image without any occlusions by the clouds or cloud shadows. Therefore, it is a much more reliable predictor of the green reflectance properties of the surface than the green band of the RGB image collected from the photographic camera (which may contain any of the problems with clouds and clouds shadows we have mentioned so far). Our aim is to find a way to use this information to correct the original RGB images to compensate for the cloud shadows. We begin by discussing the LIDAR image in a bit more detail.

2.2 Straight-Forward Replacement of Green Band With LIDAR Data

At first glance, it might be tempting to try to use the LIDAR reflectance data to simply replace the green band in the RGB triplet of each pixel in the photo mosaic. Unfortunately, even after scaling the LIDAR data to the same range (0 to 255 or 0.0 to 1.0) as the RGB

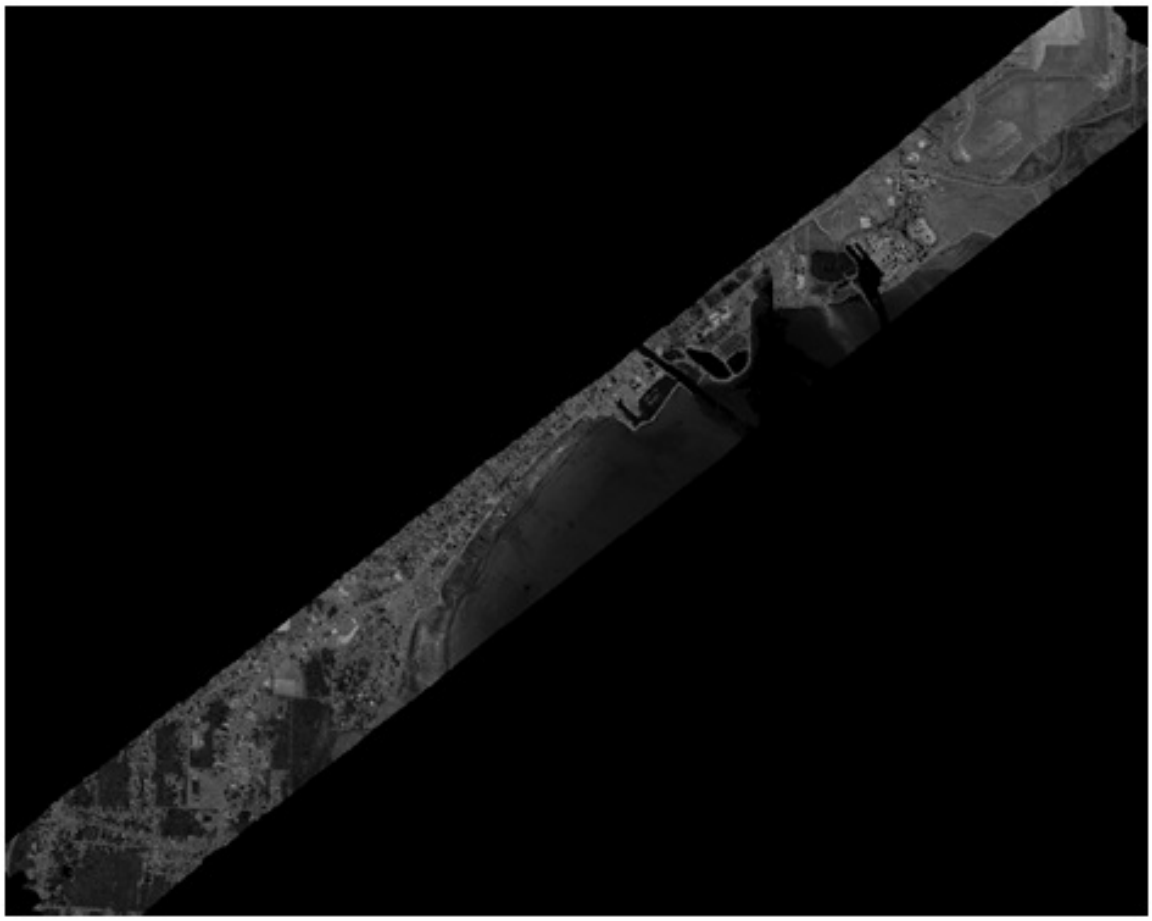

Figure 2.1: Original LIDAR reflectance image.

data, the straight forward replacement produces adverse affects and therefore is not a good option. As can be seen in Figure 2.2 this results in an image which is dominated by this green replacement. We could scale it back further, but multiple attempts at this produced nothing valid. It is obvious that simply replacing this value will not work effectively.

This is primarily due to the high level of correlation between the 3 axes of RGB. For example, if a pixel has high levels of red and green, say 225 or more, then the blue level should also be high, somewhere relatively close to 225. Others have shown that there are difficulties with this (color transfer guy), and have even proposed new, more highly de-correlated color spaces in which to work for this reason.

Therefore, our initial plan was to use a non-linear relationship to model the ratio between

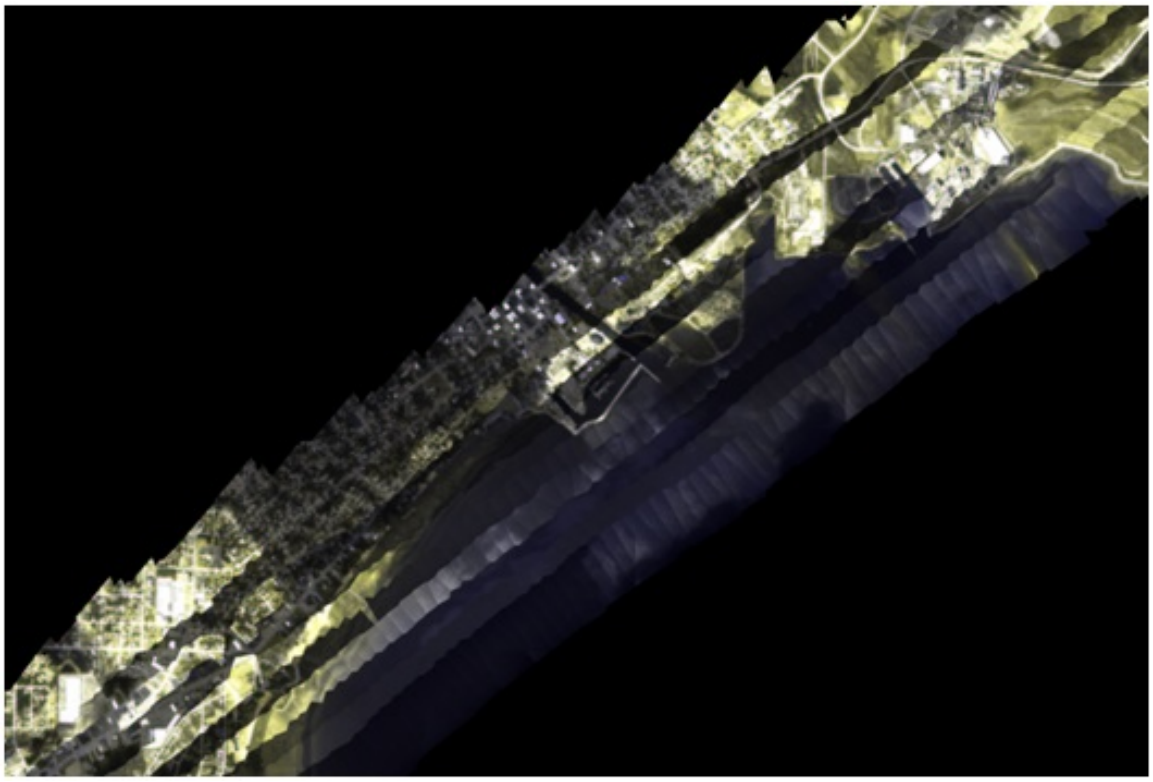

Figure 2.2: Mosaic created by replacing the green band of each RGB triplet with its corresponding LIDAR value.

intensities in the green band of the RGB image and the LIDAR data. Because the LIDAR data and RGB is acquired and provided to us at different resolutions, the data needed to be adjusted to account for this. The LIDAR data is floating point data and has a range of intensity of about 0 to 2.0 (actually, this is the range of the data provided and not the range of the equipment, which was not provided), and the LIDAR reflectance image has a resolution of 4 meters per pixel. The RGB pixel data has the typical integer range of 0 to 255 per color band, and each pixel has a ground resolution of 20 centimeters.

Therefore, when building a non-linear relationship between the two, we first scale both datasets to the same range, and then we compute the local average of the RGB data such that we get a single intensity value that represents the same ground coverage area of a LIDAR pixel. However, we only need to do this for the green band of the RGB image, as this relationship with the LIDAR (green laser) is more valid due to its wavelength similarity.

This results in a local green averaging kernel filter window size that is 20 times larger than a single pixel in the LIDAR data, giving us the average green color for approximately the same surface coverage.

In addition to the dynamic range scaling and resolution scaling above, there are additional steps which must be taken with the LIDAR data. Unfortunately, the LIDAR data provided is very noisy, with frequent pixels which have no data returns. These deficiencies are due to the limitations in the current version of the CZMIL data acquisition instruments, and will hopefully be upgraded and enhanced in the future.

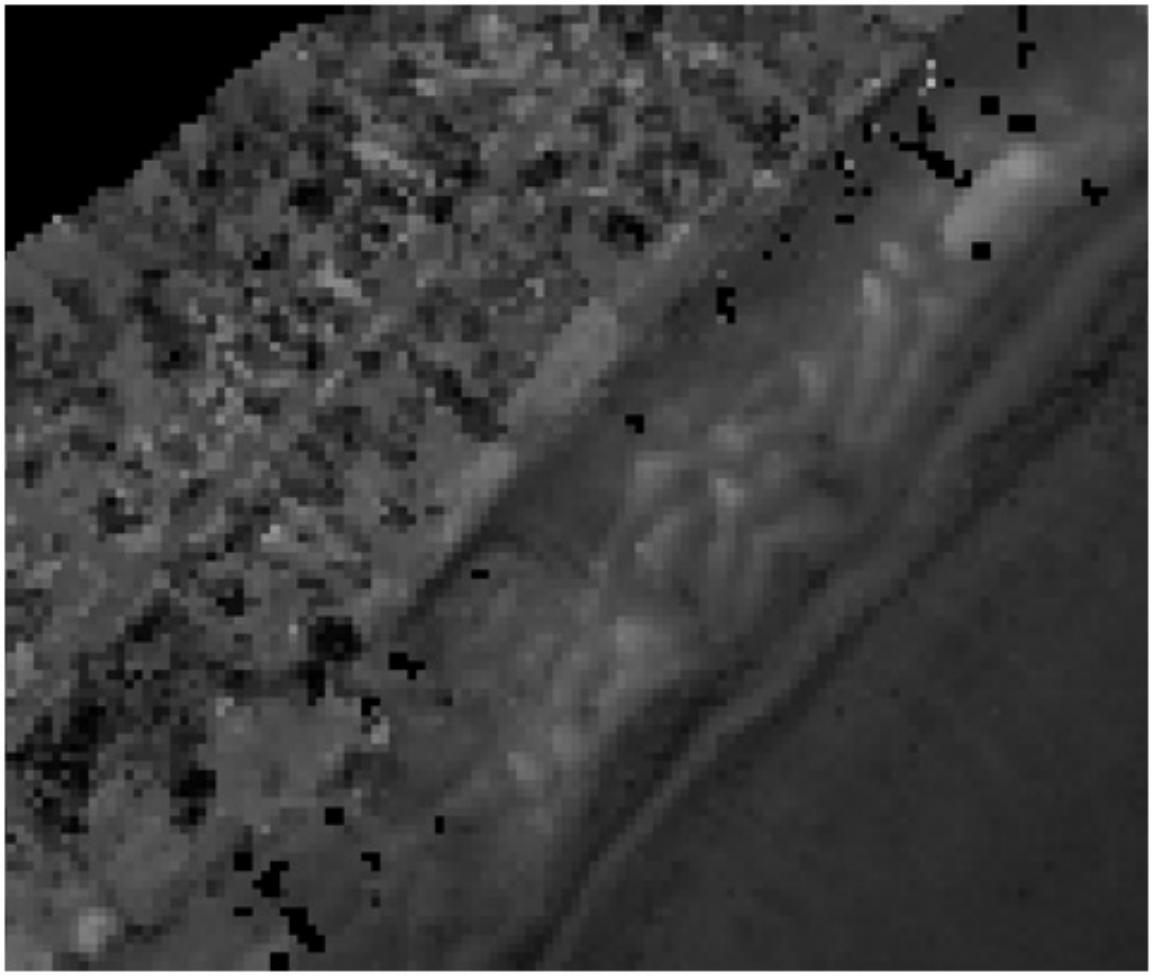

Figure 2.3: LIDAR data with missing return data. Notice the black holes with 0.0 intensity which represent missing data in the LIDAR reflectance image.

2.3 Filtering the LIDAR Data

As it stands, however, we must make some correction to the data. The obvious choice is to filter the data in two passes: a first pass which removes all data with intensity values of 0.0 where appropriate, as this simply means there was no data returned for this pixel. For this, we simply take the local average of the nearby surrounding pixels, and replace the pixel in question with this value. As long as it is only one or two pixels with values of 0.0 we can assume that this is a reasonable correction. For the current purposes of this research, we are only considering the land areas, as the data we have for the water bodies is incomplete. We would like to have the data from the water-penetrating LIDAR to perform the color corrections discussed ahead over water as well. However, we expect there to be many additional difficulties due to the different properties of water, including waves, wave caps, and reflectance properties. Therefore, our lower bounds (high-intensity-pass filter) is set to filter anywhere there are pixels with intensity values of 0.0 within the main image, and the results in water bodies may be ignored. The same close-up of the image after performing this lower bounds filter is shown in Figure 2.4.

A second filter pass of the LIDAR data is then done to de-speckle and smooth the LIDAR image, as there is still a lot of noise present in the data in the form of high variance of intensity between pixels, as seen in the previous example image. This distortion will bleed through to the final image mosaic if not handled beforehand. Using a mean kernel filter, we are able to smooth these variances, while retaining the detail of the edges on buildings and roads. The result of this filter produces an image as seen in Figure 2.5.

Once this process is complete, we now have relatively clean LIDAR data to work with to build a relationship between it and the RGB data. Initially we attempted to use a naive approach based on the simple non-linear relationship shown below. However, there are a number of problems with this approach as you can see in the next figure.

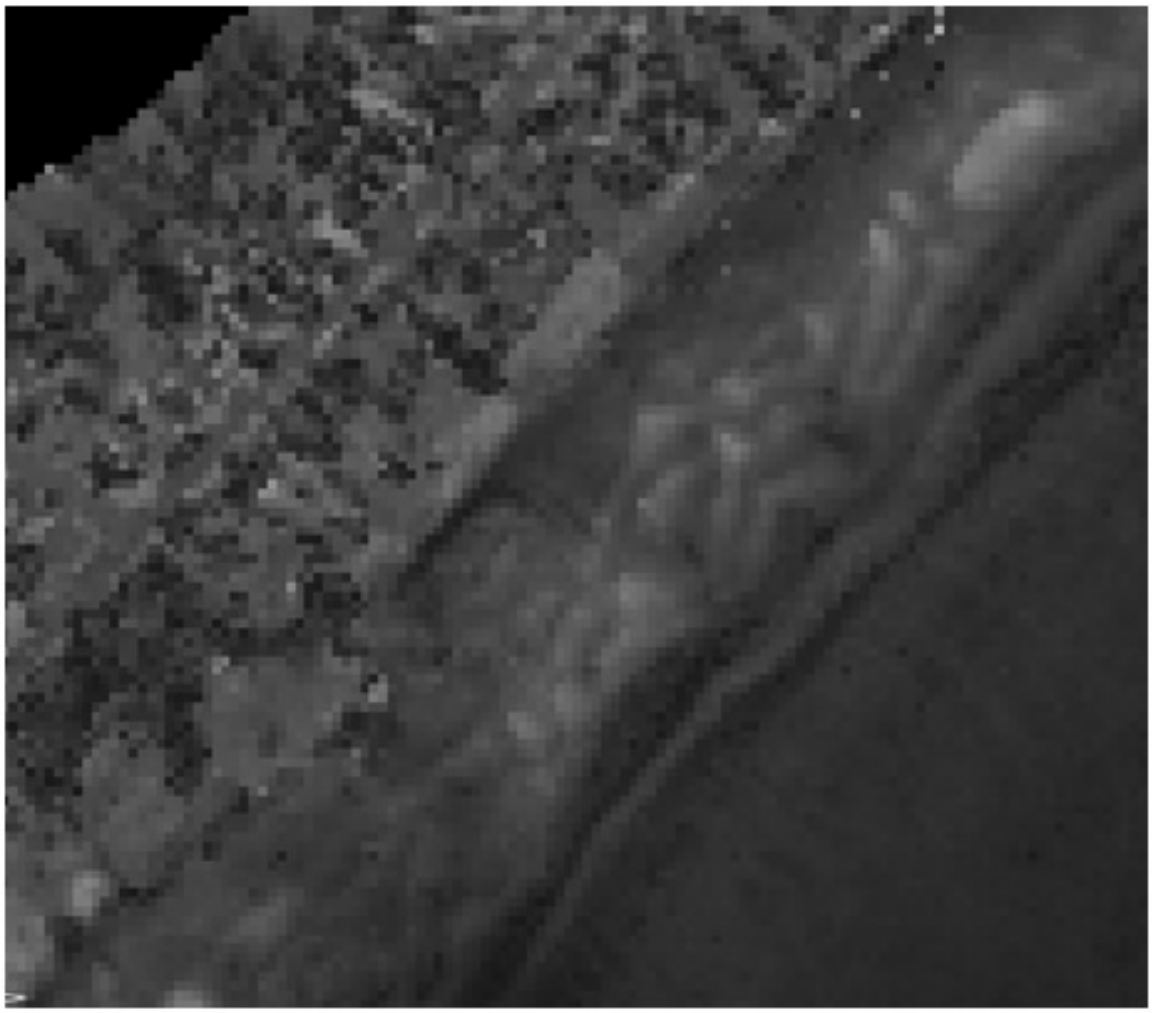

Figure 2.4: LIDAR data after doing a lower bounds filter to smooth any missing values.

2.4 LIDAR to Green Ration - Scale Factor

As you can see in the following equation, we compute a scale factor by first normalizing the two data points and then divide the green average by the local LIDAR average. Here, LIDAR average simply refers to the fact that the LIDAR was smoothed using an averaging filter (of which the kernel size is configurable).

$$sf = LIDAR_{avg}/green_{avg} \tag{2.1}$$

Once we compute the scale factor, we need to use it to modify the original image which

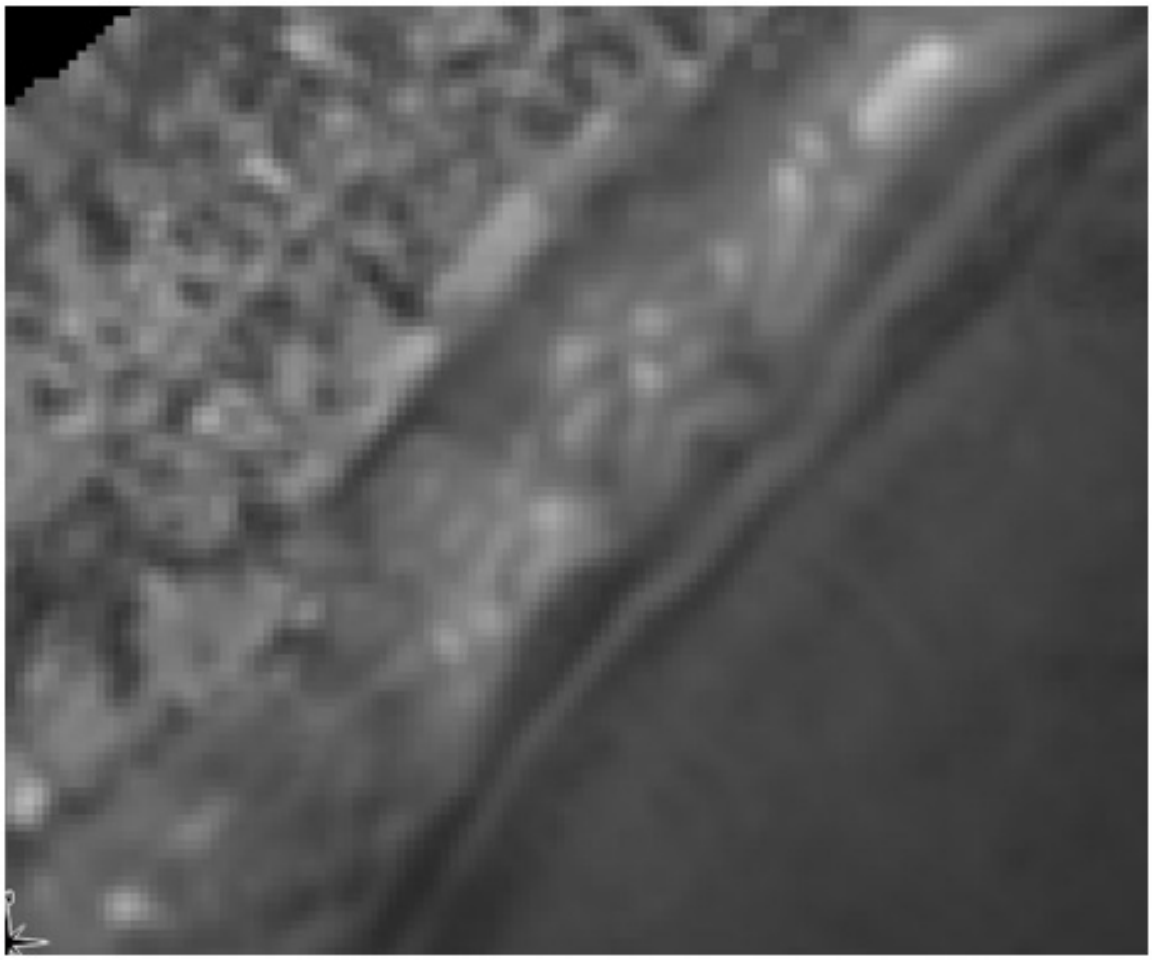

Figure 2.5: LIDAR data after smoothing with Gaussian filter.

may or may not contain cloud. If a pixel in the original image is under cloud shadow, then $green_{avg}$ should be a relatively low value, and $LIDAR_{avg} > green_{avg}$ should be true. This will produce a scale factor which is higher than 1.0, which when multiplied by the original RGB triplet will have the effect of raising the intensity to compensate for the dark cloud shadow. In the opposite case, when we have an area which is overly bright due to some surface's highly reflective properties, we will pull its intensity down to produce a more even-looking intensity distribution. Applying this technique results in the image in Figure 2.6.

Figure 2.6: Full view of mosaic after applying scale factor correction.

2.5 Scale Factor-Based Results

The results obtained using the scale factor are quite good, as they succeed in producing an image which has a rather consistent intensity throughout. Still, there are a number of problems with this approach. In Figures 2.7 and 2.8, we have two close-ups of the same area, one from the original mosaic, and one from the scale factor corrected image. Notice how the highly reflective asphalt parking lot in the upper right hand corner gets grayed out in the corrected image. One can also see this in the road surface to the bottom left of the images. We could minimize this effect by only applying the scale factor where there are cloud shadows present.

Unfortunately, we do not know exactly which pixels are obscured by cloud shadow from the scale factor alone. However, we could approximate this globally by assuming that some threshold value of the scale factor could be a deciding factor as to whether or not the pixel

was in cloud shadow or not. The value $sf = 1.0$ would be a good candidate, as it is the pivot point at which we either raise or lower the intensity of the current pixel. However, this does not work in practice, as there are multiple combinations of green and LIDAR pixel data which result in scale factors that may also produce values of $sf = 1.0$, even though a pixel is not in cloud shadow. It simply is not globally reliable enough to correctly predict whether indeed a pixel is in cloud shadow or not. While the scale factor is a good way to equalize the intensities in the mosaic, it does not produce a model which truly relates to the pixel's status as in cloud or not cloud.

Figure 2.7: Close-up of original mosaic (with cloud shadows).

Figure 2.8: Close-up of mosaic after using the scale factor to modify the image.

Chapter 3

CLASSIFICATION

3.1 Introduction to Classification

After initial attempts at a simple, straightforward non-linear relationship between LIDAR and RGB proved to be problematic, we set out to find a better method for identifying cloud shadow and correcting the color of the images in the mosaic. It quickly became apparent that nearly all of our promising ideas depended on knowing whether a pixel or group of pixels were either in cloud shadow or not. This is essentially a classification problem, in which we are given red, green, blue, and LIDAR pixel intensities as input and we expect a binary classification, cloud or not cloud, as an output. Once we know the answer to this question for every pixel, we can then begin to perform more intelligent modeling of the color transformations.

Various classification and regression models exist which are found in the literature. Decision trees, rule-based classification, neural networks, linear classification, and support vector machines are just a few examples of these algorithms, each with their own advantages and disadvantages. We chose to focus on two models: linear classification (using the LIBLINEAR [6] library) and support vector machines (SVM - using the LIBSVM [1] library). As will be discussed in a moment, ideally we would like to choose SVM as the classification algorithm, but due to the lack of any robust parallel implementations, the training time on very large data sets can be a problem, as well as the computational complexity of testing a new data point against the SVM, which is a factor of the number of support vectors. Although we do achieve some level of parallel performance with LIBSVM (due to data decomposition across our already parallelized multi-process software), the very

large number of pixels that must be classified against the thousands of support vectors still results in an unacceptably long run time.

Therefore, we have also investigated using a linear classifier, as it is extremely fast in comparison to the SVM, and has comparable accuracy. However, the SVM is much more tolerant to noise in the training data than the linear classifier, and as stated previously, we are dealing with very noisy data in the first place, so this is a large benefit. Also, SVMs are capable of dealing with non-linear relationships between their attributes, allowing them to more accurately model the expected non-linear relationship between the input attributes in this research. SVM's are a very active topic of research, and hopefully we will see more efficient, scalable parallel algorithms developed in the near future. Nevertheless, we begin our classification discussion with the linear classifier.

3.2 Linear Classifier

The software used to perform the linear classification within our software is called LIBLINEAR. While it is primarily designed for data whose attributes are linearly independent, when given enough number of samples, it can be shown that the linear classifier can perform nearly as accurately as the SVM classifier, with a fraction of the computational complexity. Indeed, with our data, we have a total sample set of nearly 45 million pixels at 0.8 meter resolution (our common working resolution due to memory constraints), and the time to train the linear classifier is many orders of magnitude (days) faster.

In order to use any classifier, we must first decide which attributes are important enough to include in the classifier for each of the data points. Also, we must determine whether we are going to use a binary (two outcomes) or a multi-class (more than two outcomes) classifier. In our case, we chose to go simply with a binary classifier, with outcomes of either cloud or not cloud. Further research could be performed in the future which might reveal whether using multiple classes such as heavy cloud, light cloud, etc. are any more helpful. However, this is beyond the scope of this project.

Therefore the, only other choice besides the type of model is which input attributes to use. In order to show the effectiveness of our chosen attributes, the following images show both the accuracy (via 10 fold cross validation of the model) and a cloud/no cloud map of each set of chosen attributes. The cloud/no cloud map shows, for each pixel in the output mosaic, whether or not the classifier predicted that the pixel was in cloud (blue) or not in cloud (red). We can compare this to the original, uncorrected mosaic visually to see how well the classifier performs. The cross-validation of the model is also a useful prediction of how accurate the model is, but since we are dealing with images and the human perception thereof, it is very useful to see the cloud/no cloud map to be even more certain of the accuracy of the predictions.

Classification algorithms are, by their very nature, supervised learning methods, and as such, they must first be built using a training set of data. Therefore, we need to provide the algorithm with a set of pixel data in which the outcome (cloud or no cloud) is known. So we set about to build these data sets manually from individual images chosen from the total set of input images. We tried to choose images which had some areas of sunlight and some areas of cloud shadow, in order to have a more balanced data set. It was also desirable to try to have as many different textures (i.e., trees, roads, buildings) that were both in and out of cloud shadow. Next, an image mask was created for each selected image in an editing program (GIMP) by drawing and filling in areas where there was cloud shadowed regions with the color red (255, 0, 0). This value was chosen as the pure red color does not occur anywhere in any of the input images, and it if had, it would most likely be at a negligible quantity. Figure 3.1 below shows a couple of the selected training images along with their masks.

Once the masks have been created, the images are run through the code to produce data files in the format for training with LIBLINEAR. For each mapped pixel with a value of (255, 0, 0) we produce a training tuple whose class is cloud, and all other pixels are marked as no cloud. In our case, we created 35 sets of training data created from 35 pairs of

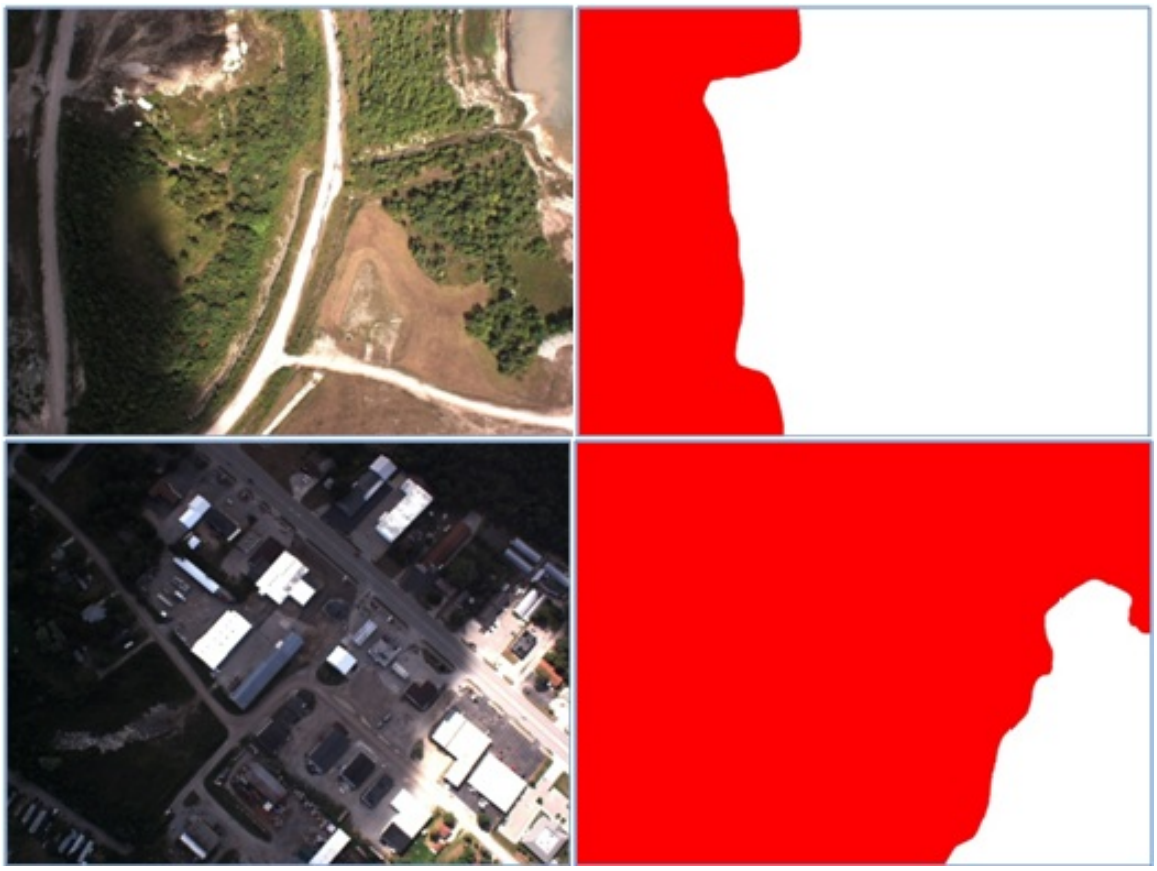

Figure 3.1: Training images. The images on the left contain cloud shadow covered pixels. The image masks on the right are created and used in the classification process.

input images and input masks. Obviously it would be desirable to produce these data sets automatically, instead of manually with image masks, but this is a limitation of supervised classification, and future efforts could be made to solve this with an unsupervised method, such as clustering. However, in our case, once the model is built, it can be used on the entire survey area and so the longer process of training is only performed once in the beginning. Subsequent mappings and modifications in the same flight mission can re-use the model. Also, it may even be useful to build training samples from many different flight missions, with the goal of creating a model which could be increasingly effective on additional surveys.

Typically in data mining applications such as classification, it is beneficial to do some sort of feature selection method to determine which attributes (or features) are most instrumental in affecting the predicted class. We performed two common types of feature selection

methods on the possible input attributes (red, green, blue, LIDAR average, green average, and scale factor): Information Gain and Gain Ratio. While both do not produce the same order of importance for each attribute, both methods did produce the same top attribute - LIDAR average. Also, since the number of possible attributes is so small, we also simply built models with all combinations of attributes, and evaluated each one of their 10-fold cross-validation percentages. The results of a couple of these are shown below.

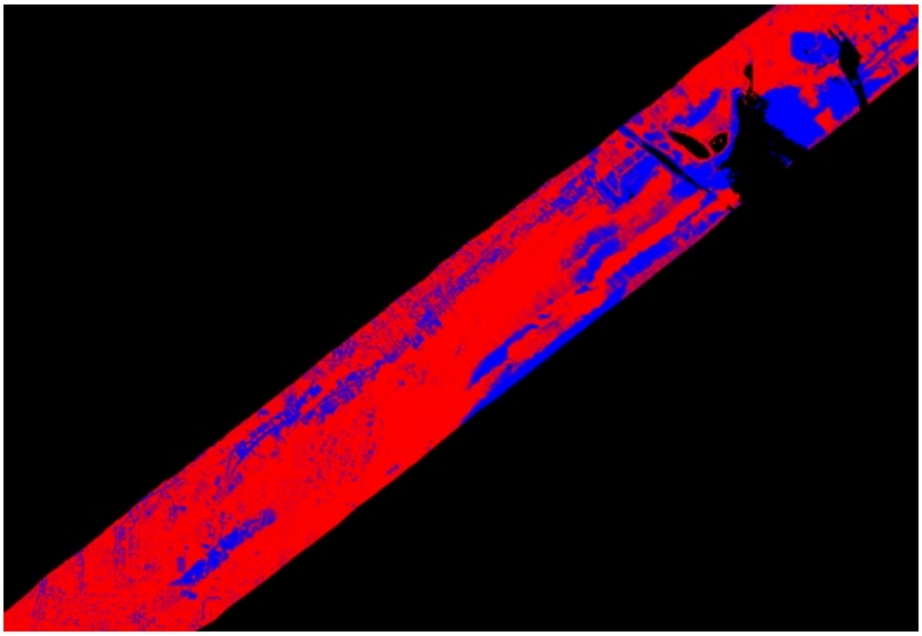

Figure 3.2: Linear classifier with RGB values only. Cross validation Accuracy = 64.7872%.

The combination of all the attributes (R, G, B, LIDAR average, green average, and scale factor) produced the highest accuracy (85.2561%). We also were able to obtain very similar accuracy without the scale factor attribute (85.2265%), whose value is simply a factor of LIDAR average and green average. With these results in mind, and the fact that there was virtually no difference in the training and testing execution times, the decision was made to omit the scale factor from consideration with LIBLINEAR as the increased accuracy was fairly negligible, and the term was non-linear in the first place.

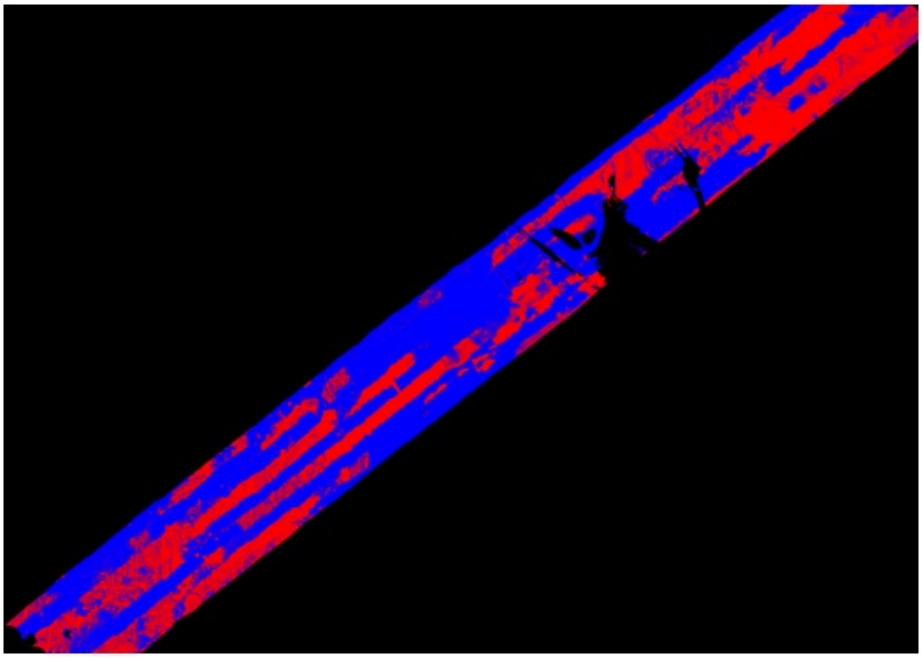

Figure 3.3: Linear classifier with RGB and $LIDAR_{avg}$. Cross validation Accuracy = 80.8666%.

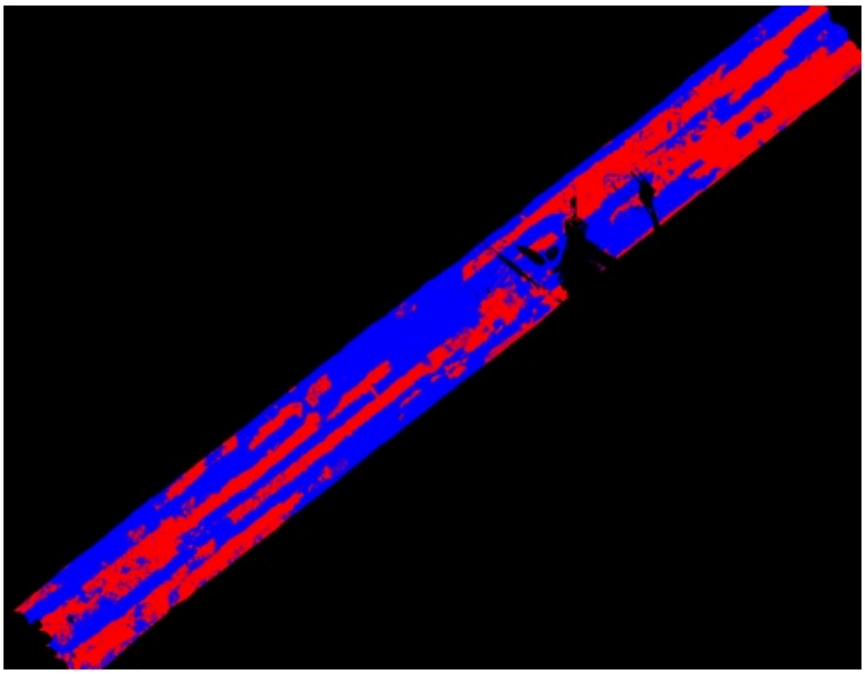

Figure 3.4: Linear classifier with RGB, $LIDAR_{avg}$, and $green_{avg}$. Cross validation Accuracy = 85.2265%.

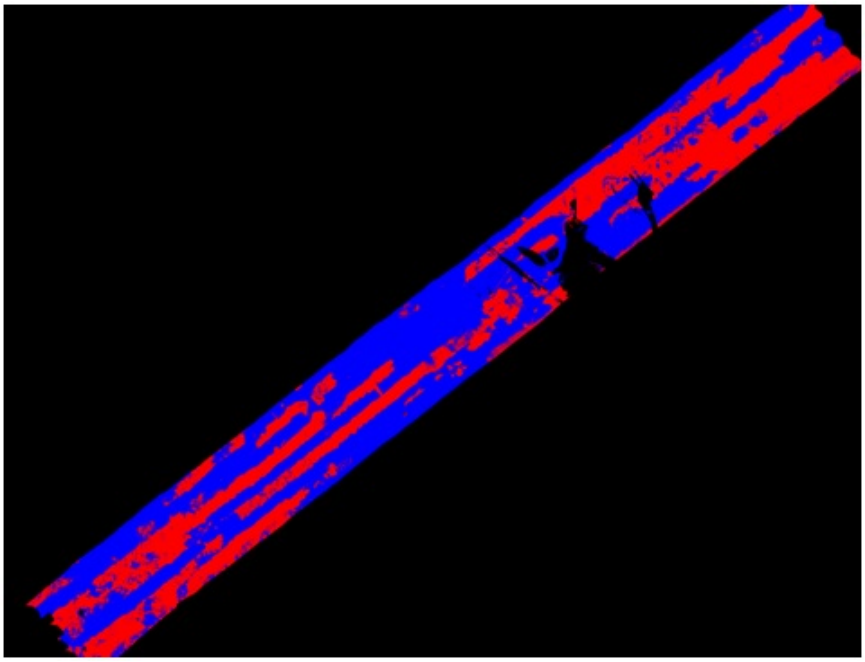

Figure 3.5: Linear classifier with RGB, $LIDAR_{avg}$, $green_{avg}$, and sf . Cross validation Accuracy = 85.2561%.

3.3 Support Vector Machines

Support vector machines (SVM) are a type of non-probabilistic classifier which aims to find the maximal-margin separating hyper-plane (MMH) between classes of data in a dataset. It is capable of handling data in which attributes are either linearly or non-linearly separable. If the data are non-linearly separable, then we use a non-linear mapping to a higher dimension, one in which a linear separating hyper-plane can be found. It turns out that the linear separating hyper-plane in this higher dimensional space is the same as a non-linear hyper-surface in the original dimension [8]. We wish to find the hyper-plane which maximizes the margin between the two classes of data points, which is very helpful in preventing a model which over-fits the data [8]. The MMH is so named because it is the hyper-plane which has the largest margin between itself and the hyper-planes which define the sides of the margin. These margin sides are defined by the training tuples which are nearest the MMH, and there may be more than one on each side of the hyper-plane. These support vectors then define

the model, and in order to use the model to test a never before seen tuple, we simply use the decision function

$$d(X^T) = \sum_{i=1}^{l} y_i a_i X_i X^T + b_0 \tag{3.1}$$

where l is the number of support vectors learned from training, yi is the class label of support vector Xi, and XT is the test tuple (is a Lagrangian multiplier). The sign of the result tells us which class the test tuple belongs to. As you can see, the complexity of testing a new tuple against the model is primarily based on l, the number of support vectors, and not on the dimensionality of the data, which leads to a model which is less prone to over fitting [8]. The support vectors are thus the most important training tuples in the training set. As mentioned earlier, SVMs also have a high degree of tolerance for noise in the training data, and was a primary reason for us choosing these algorithms.

We chose LIBSVM as our code base for implementing various SVMs within our software. Other implementations exist, including parallel implementations such as PGPDT, but LIBSVM provided a complete toolset for sampling, parameter selection, and using multiple types of SVMs on the data, such as C-SVM classification, ν-SVM classification, C-SVM regression, and ν-SVM regression. Overall, it is a very well rounded SVM library. We built many models during the course of testing, and results of a select few are shown ahead. LIBSVM can also handle the case when the data are simply linear, but the library is not primarily used for this, and therefore the author suggests LIBLINEAR for such cases, as the execution time is much faster [12]. However, for the non-linearly separable cases, LIBSVM provides a number of choices, including polynomial kernels, radial basis function (RBF), and custom kernels. The kernel function is what is used to non-linearly map the data to a higher dimension. We evaluate the execution time and classifications accuracy results below. Although LIBSVM was used to perform linear classification on the data as well, we found that indeed LIBLINEAR was much more efficient in training, testing, and accuracy than the linear functions in LIBSVM. Nevertheless, we include the results from each of them for

completeness.

Unlike the linear classification described in the previous section, the time to train the SVM with 45 million datasets is not practical (on our system, nearly two weeks). Therefore, the models below were built using a 220,000 sub-sample of the data set. The samples were chosen using stratified random selection, ensuring that the training data will be well balanced. Although the sample is only roughly 0.4% of the total data, the models built from this data are very accurate.

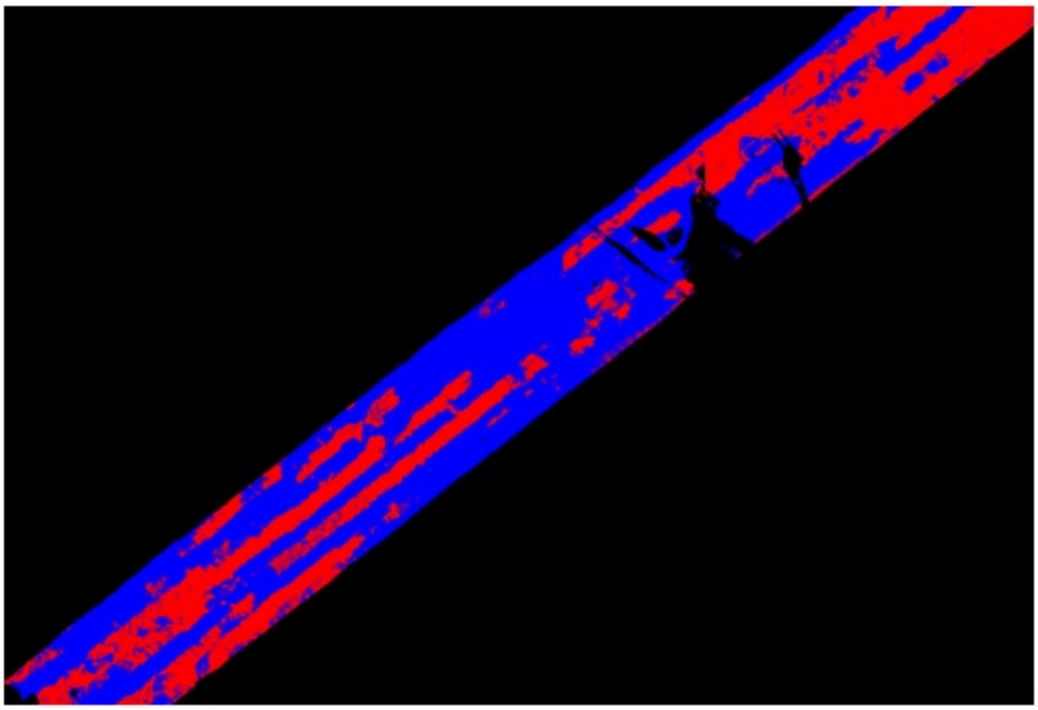

Figure 3.6: C-SVM Linear Kernel. 10-Fold cross-validation = 84.9009%. Based on 220,000 samples.

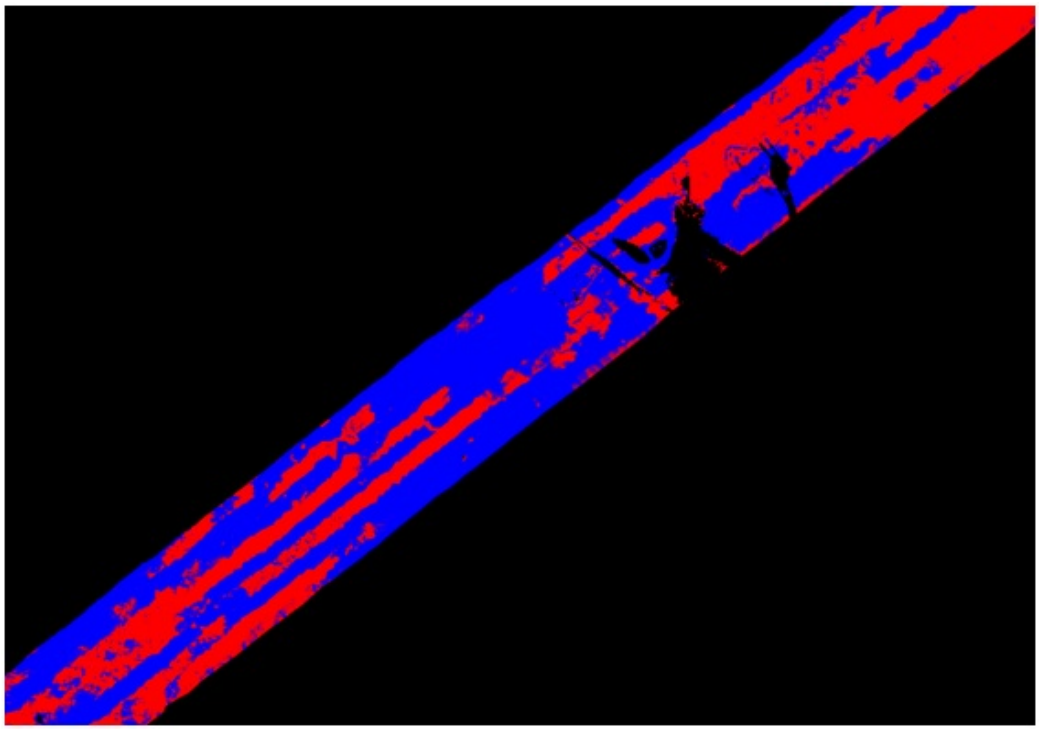

Figure 3.7: C-SVM Quadratic Kernel. 10-Fold cross-validation = 85.3536%. Based on 220,000 samples.

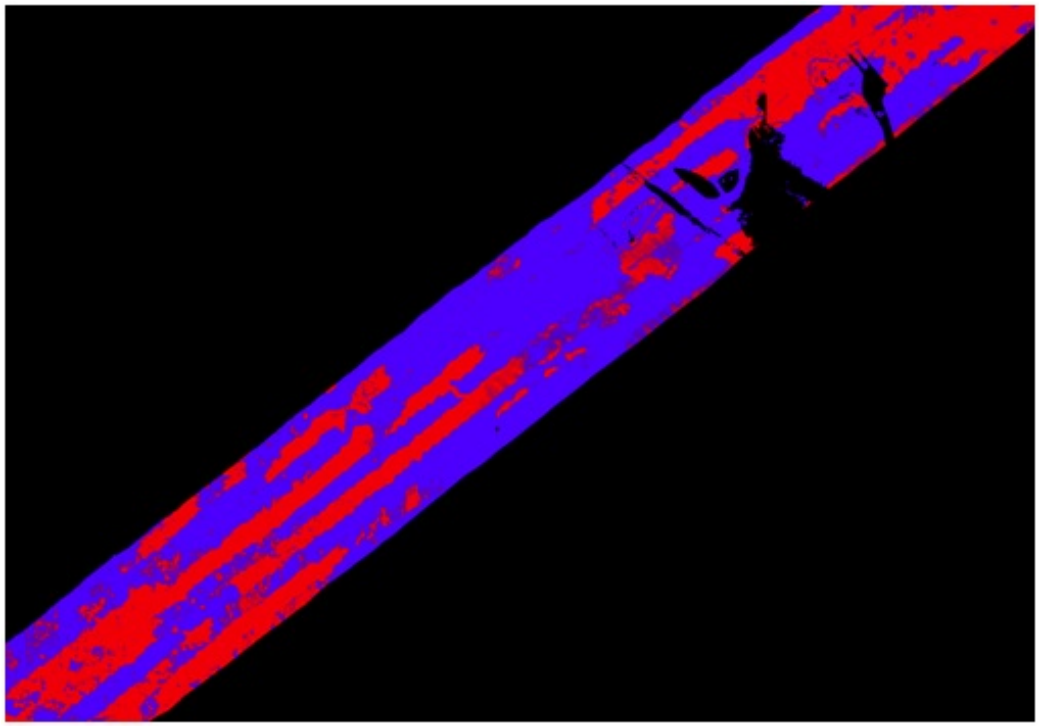

Figure 3.8: C-SVM RBF Kernel. 10-Fold cross-validation = 88.21%. All attributes used. Based on 220,000 samples.

Chapter 4

COLOR CORRECTION

4.1 Introduction

As can be seen from the previous classification section, we have been largely successful in detecting cloud shadows within the image mosaic. However, we still have the more difficult task of correcting the color of the cloud shadowed region in order for it to appear as close as possible to what it would have looked like had the sunlight not been obscured by a cloud. Also, we want this corrected region to appear seamless to its surrounding, non-cloud covered areas. As mentioned in some of the background material, there are a few different techniques for solving similar color matching problems. We briefly discuss a few of these basic methods in this section.

4.2 Histogram Equalization

Histogram equalization is the foundation for histogram matching, discussed in the following section. It is most commonly used to enhance the contrast in an image, spreading its histogram more evenly across the dynamic range of an image [7]. This is interesting, because generally in a cloud covered image, there exists a large count of low intensity values in each band of the RGB image. This is easily seen in Figure 4.1.

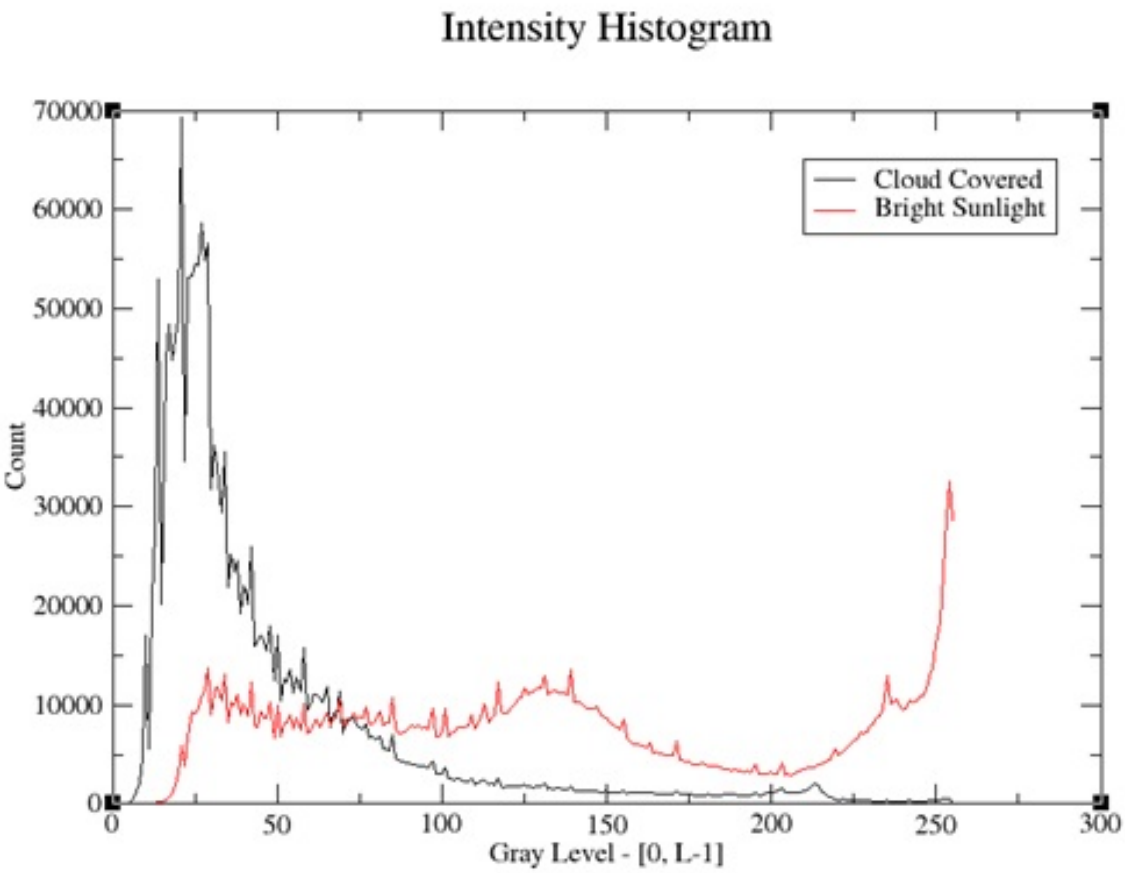

Figure 4.1: Histogram of two images, one with majority cloud shadow (black line) and one with majority sunlight (red line). Note the peaks in the cloud shadow covered image.

Notice the large peaks near 0 (the darker ranges) in the cloud shadow covered image. Adjusting this by spreading the intensities across the dynamic range of the image will remove this large spike in the histogram, and will result in an image that does not appear to be darkened by cloud. Straightforward histogram equalization was used on an image which was mostly cloud covered, and equalizing the image resulted in a brighter image as seen in Figures 4.2 through 4.5.

Figure 4.2: Source image for histogram equalization.

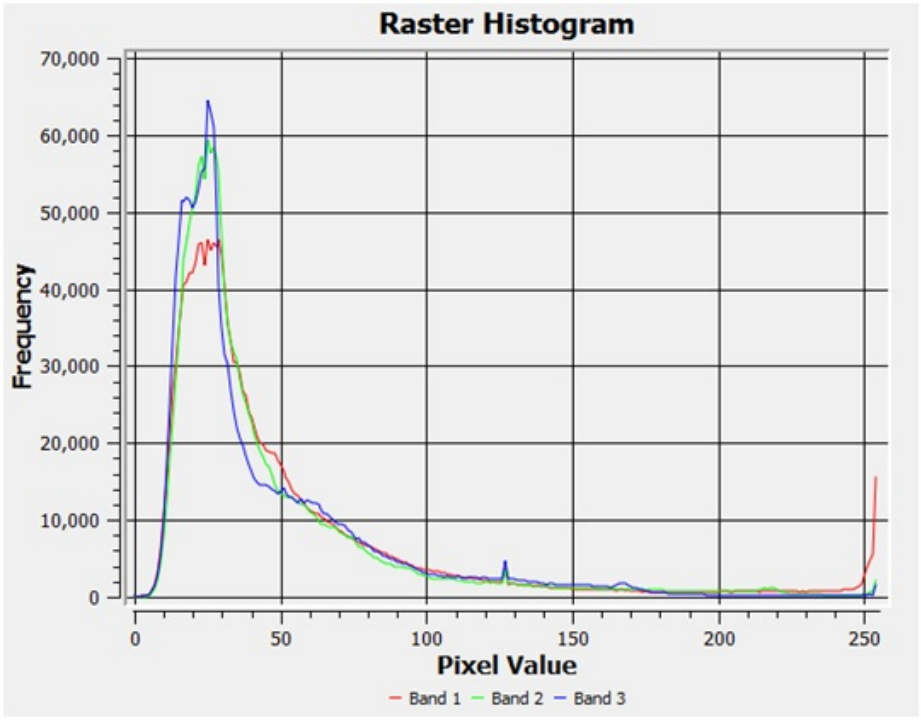

Figure 4.3: Source image histogram.

Figure 4.4: Histogram equalized image.

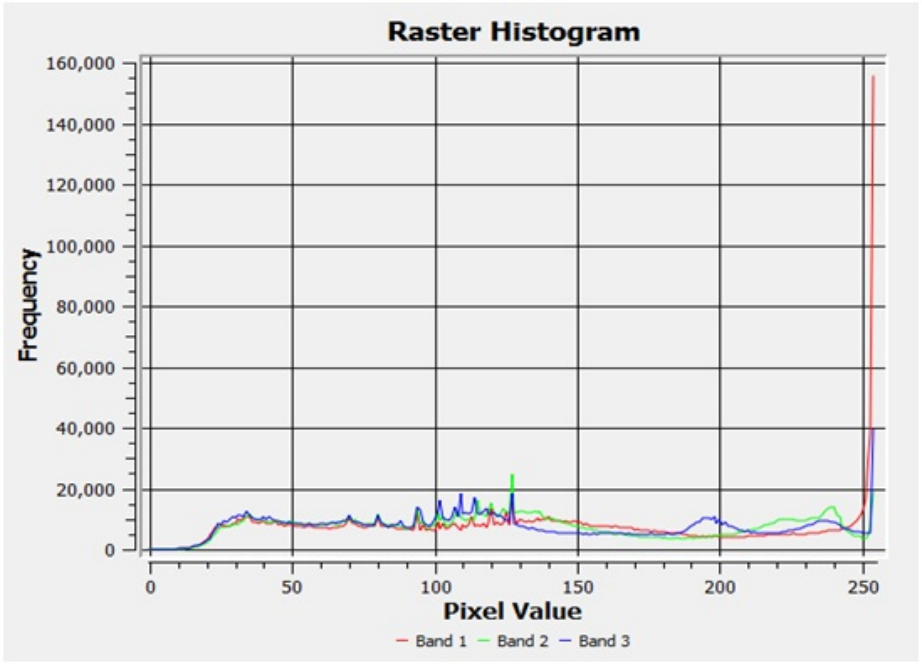

Figure 4.5: Histogram of equalized image.

This is essentially what we want to do to a cloud covered image, is increase its intensity, but as can be seen from the above results, equalization washes out much of the color of the original image, altering the distribution of intensity dramatically. This could be applied to the entire mosaic as a whole, but the loss of the original color distribution is not satisfactory. We instead aim to have the color distribution take on that of a normal, cloud-free color distribution. The next section discusses histogram matching, which extends on this idea of equalization to also take into account information from a target image.

4.3 Histogram Matching

Histogram matching is an attempt to match one image's intensity distribution to another's. In this way, we hope to take an image with cloud shadow, compute its histogram in each color band, and map it to the histogram of an image with cloud-free scenery [7]. In general, this statistical distribution transformation is an improvement over simple histogram equalization, yet it also has its problems. Figure 4.6 through Figure 4.11 show a set of three images and their histograms: The first image is the original cloud covered region. The second image is the target image (i.e., the cloud-free image whose distribution we would like to transform to), and the third is the final, processed histogram-matched image.

Figure 4.6: Source image for histogram matching.

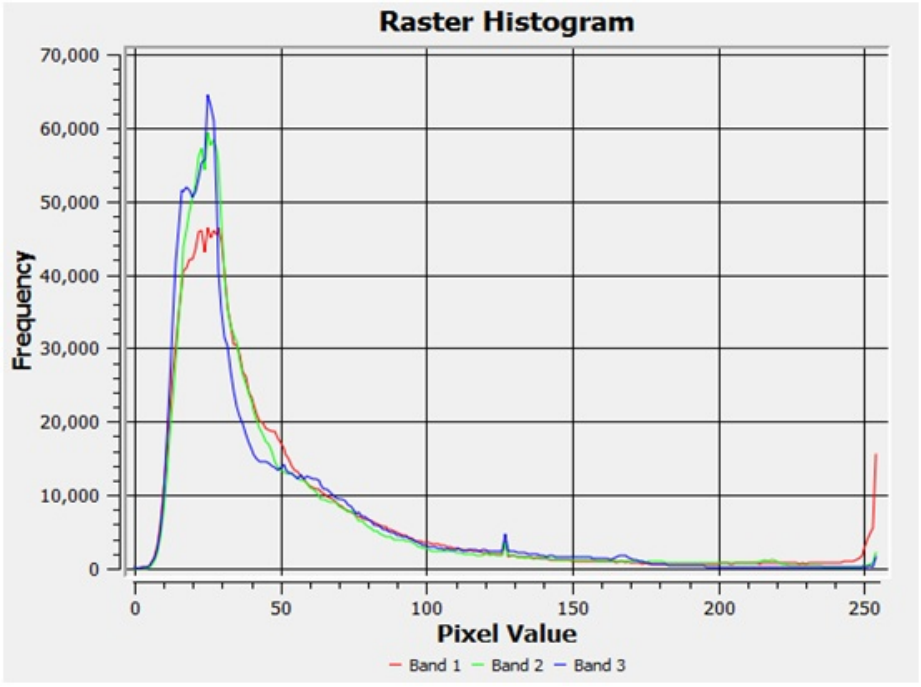

Figure 4.7: Source image histogram.

Figure 4.8: Target image for histogram matching.

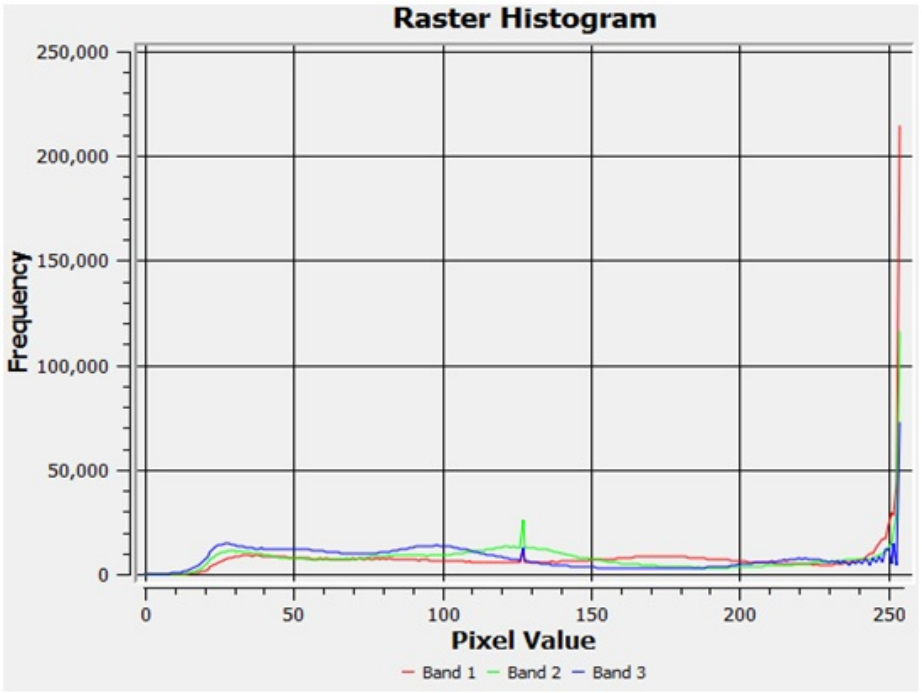

Figure 4.9: Target image histogram.

Figure 4.10: Histogram matched image.

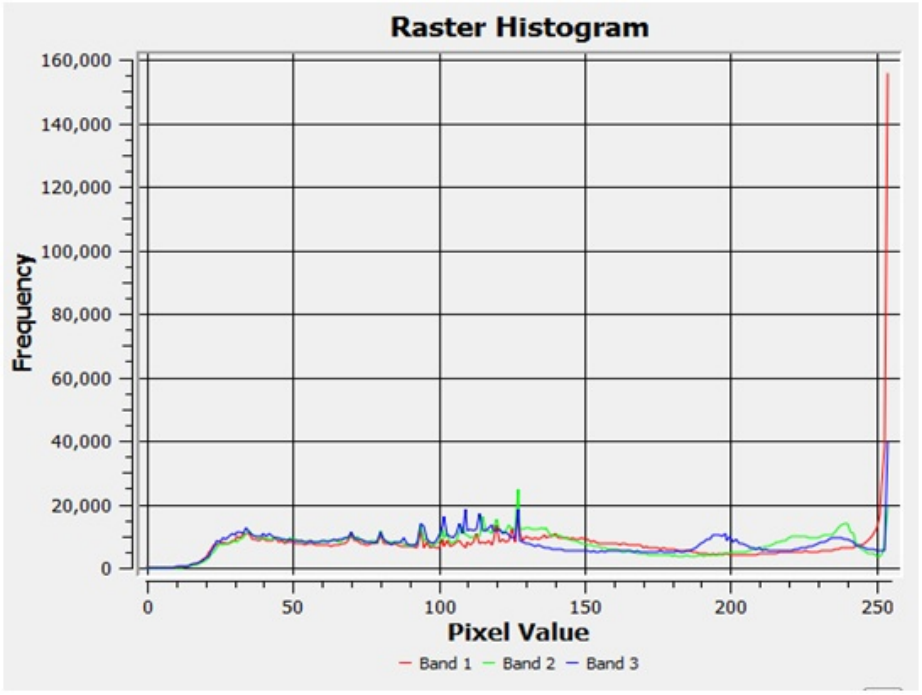

Figure 4.11: Histogram of matched image.

As can be seen, the overall cloud effect is removed, but the images still have the same washed-out look as the histogram equalization. This is primarily due to the fact that histogram matching first involves equalizing the histograms of the source and target images before doing the matching. Another problem occurs in the upper left quadrant of the photographs. The target image does not have the exact same surface features because it is a different image from the source image (as we do not have the exact same image with no cloud, or we would have used it in the first place). Although the two images to be matched can be from any area of the mosaic and in any orientation, the source and target image need to have nearly the same material distribution in the image or they will not be good candidates for histogram matching. The same problem occurs with the color transfer algorithm discussed next, but to a lesser degree.

4.4 Color Transfer

Color transfer is similar to histogram matching, in that it attempts to cause a source image to take on the color intensity distribution of another, target image resulting in a third, processed image. The difference is instead of using the cumulative distribution function as in histogram matching, the color transfer algorithm introduced by Reinhard in [21], first transforms the color spaces of the source and target images into an $L\alpha\beta$ color space. In $L\alpha\beta$ color space, the three axes, L, α, and β are highly de-correlated from each other, allowing us to manipulate the bands individually without introducing the unintended effects which arise from using RGB data. Once in $L\alpha\beta$ color space, the mean and standard deviation of the source and target image are computed, and the final image is composed by applying the mean and standard deviation of the target image to the data points in the source image. In this way, the final image now contains a similar distribution as the target image. The final step is to transform the image back to RGB color space using the inverse transformation applied earlier, which results in the final image. We implemented and tested the color transfer algorithm on the same images previously used during histogram matching in order

to compare their performance. The results of this process are listed in Figure 4.12 through Figure 4.18.

Figure 4.12: Source image for color transfer.

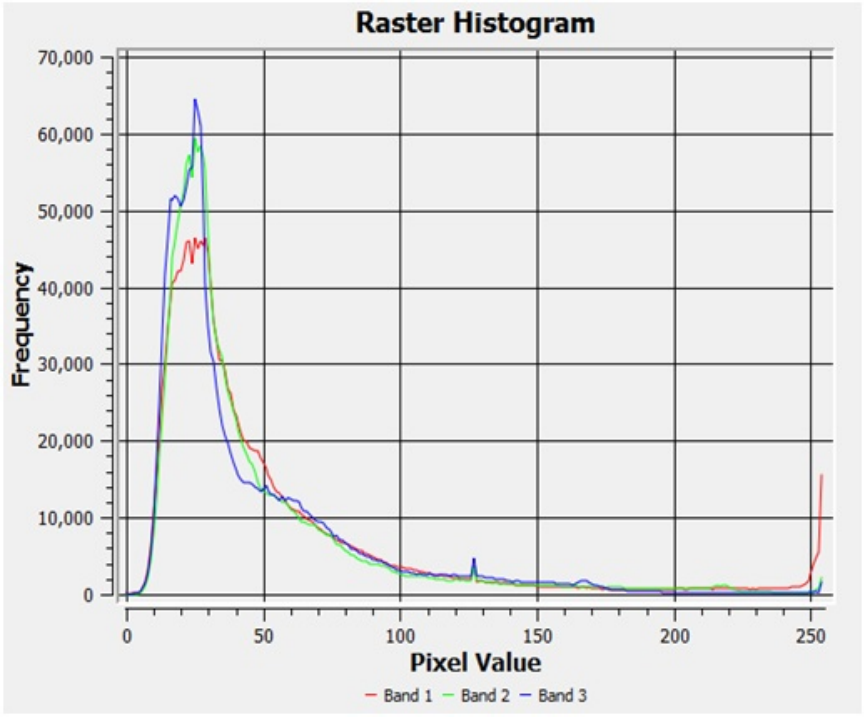

Figure 4.13: Source image histogram.

Figure 4.14: Target image for color transfer (the image we want our source image to look like).

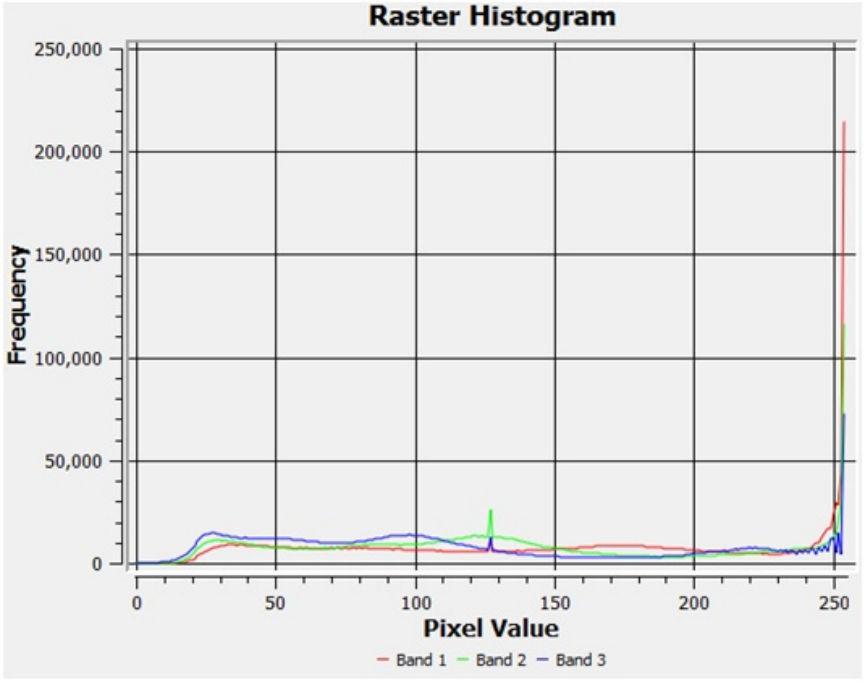

Figure 4.15: Histogram of target image.

Figure 4.16: Image created from color transfer process.

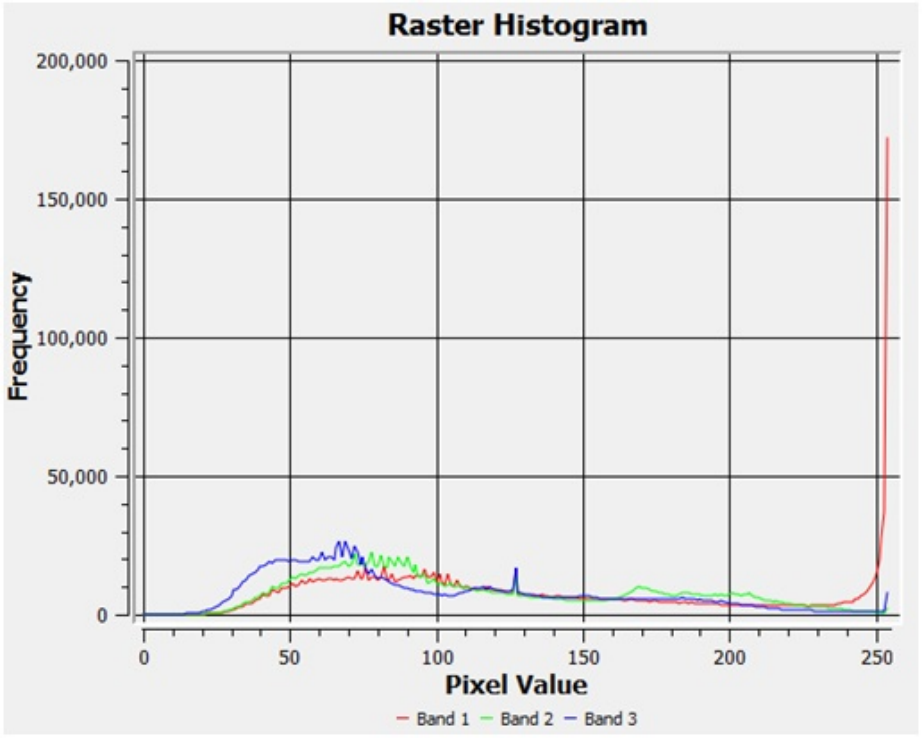

Figure 4.17: Histogram of image created from color transfer.

Figure 4.18: Histogram matched image and color transfered image, together for comparison. Note the richness of the color in the image created from color transfer.

Clearly the color transfer algorithm produces a resultant image whose color distribution is much more similar to the target image than with histogram matching. For example, the greens in the trees better match the greens in the trees in the target images, and the road and ground colors match the target image more closely as well. The upper left quadrant in this image, however, is still a problem. In the source image that we wish to modify we have a large portion of beach, which is not present in the target image. Although the algorithm is rather robust when handling different scenes with similar content (i.e. the neighborhood areas are different neighborhoods but they color match well), there is a limit to how much the content of the images can differ and still get a meaningful answer. Therefore, any global solution using color transfer would have to take this into account by ensuring that the source and candidate images contain similar distributions of data.

Note that in histogram equalization, histogram matching, and color transfer, the experiments detailed above were between single, hand-picked examples. It remains to be seen how one would integrate this globally among the candidate images for the mosaic. An image similarity method would need to be developed in order to guarantee that the transfer between two image distributions would be appropriate. Source and target images may be of different types of areas, different land cover, or alternate orientations, and, as such, make it is impossible to use pixel-for-pixel correspondence measures. Possibly some form of histogram distance measure might be useful in detecting the difference in content distribution, but this has not been explored currently.

Chapter 5

SHADOW TO SUNLIGHT TRANSFORMATION

5.1 Introduction

Since the previous attempts at shadow correction were not adequate for our purposes, we set out to find a transformation that could work globally with the entire image mosaic, and the classification methods described before were the foundation for this work. By developing a model which can predict whether a pixel is in cloud or not in cloud, we can operate on each class of pixels independently, and try to build a mapping between cloud covered and sunlit pixels. For the majority of this work, we utilized LIBLINEAR for the classification tasks, as it was similarly accurate, but more importantly, orders of magnitude faster than LIBSVM for our very large samples of data. For each model we build, we also build a classification map, which simply is an image where every pixel is either blue if it is in cloud shadow and red if it has been classified as not in cloud. This allows us to compare the accuracy of the classification process with the original, cloud covered mosaic visually, in addition to the accuracy estimate given by cross validation of the model generated.

5.2 Probability Estimate as a Function of Cloud Cover

LIBSVM provides a way to build into the model a method for computing probability estimates when classifying test sets. If the model predicts that a pixel is cloud covered, it can also give its probability (confidence) of that prediction. LIBLINEAR does not currently have this feature, so the discussion in this section is only relevant to LIBSVM. Initially we attempted to use this probability to modify the RGB triplet of the cloud covered pixels in a straightforward linear combination. However, we were not able to obtain any satisfactory results. There was no clear way to use the probability as a measure of how thick the cloud

shadow was, only whether or not the cloud was in fact in shadow or not. This leaves us to conclude that the probability measure is not a good measure of the darkness or lightness of a cloud shadow.

5.3 Using Scale Factor Only in Cloud Shadow-Covered Areas

Once the classification model is built, it can then be used to modify only the areas of the mosaic which are predicted to be in cloud cover. Since we also know which pixels are not in cloud cover, we can leave them untouched, retaining their original information. Unlike the scale factor method discussed in section 2.4, we do not need to find a threshold which defines cloud and not cloud. We simply have two possibilities for the classification, and can therefore treat each one distinctly. However, the scale factor still has value when the pixel is predicted to be in cloud shadow, and using it in the same way as when we applied it to the entire mosaic, results in the image in Figure 5.1.

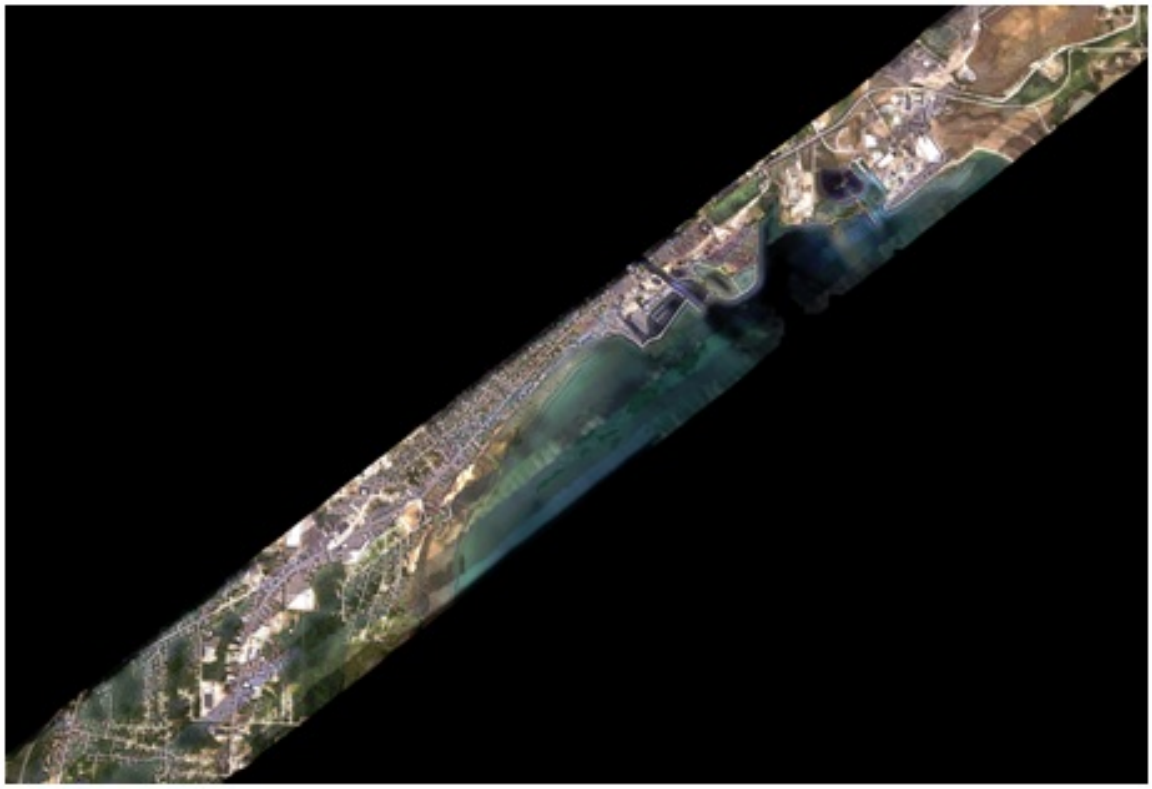

Figure 5.1: Mosaic produced using scale factor only in areas classified as cloud shadow.

Figure 5.2: Closeup of mosaic in Figure 5.1.

Figure 5.3: Close-up of original mosaic where scale factor was used on all pixels (for comparison to Figure 5.2.

Figures 5.2 and 5.3 above are a close-up examples of the differences between using the classification to modify only the cloud covered pixels, and using the scale factor across the entire image. Although the scale factor does produce an image which has been brightened in the cloud shadow covered areas, it still does not appear to match the sunlit areas. A better transformation function needs to be built which can more closely model the effect of the cloud shadow on the original picture. It may be a matter of preference as to whether or not to try to correct the overly bright (possibly overexposed) areas of the image. We have chosen to keep the parts of the mosaic which have not been occluded by cloud shadow as original as possible, with the intent that any information contained would be better served by not modifying it at all.

5.4 Determining a Better Transformation Function Using the Sunlight Data

In an effort to produce a better transformation function, we decided it would be best to try to model the relationship of pixels which were not in cloud shadow (sunlit) and their LIDAR data. Knowing what values of LIDAR data correspond with a given type of pixel which does not have cloud shadow present provides us with a basis for modifying cloud shadow covered pixels. Therefore, we produced an output file which contained the $green_{avg}$ data and the $LIDAR_{avg}$ data for each pixel which was classified as not in cloud shadow. After the file is created, we ran this new data file through a regression tool to fit an equation to the data. As you can see in Figure 5.4, the data has a general linear tendency, and fitting the data with a polynomial produces nearly the same curve. The resulting equations for linear and quadratic fit are listed below as well.

The resulting equation for the linear fit of the data is:

$$y = 2.9267 + 661.47x \tag{5.1}$$

The result of the quadratic fit of the data is:

$$y = -676.25x^2 + 939.12x - 21.194 \tag{5.2}$$

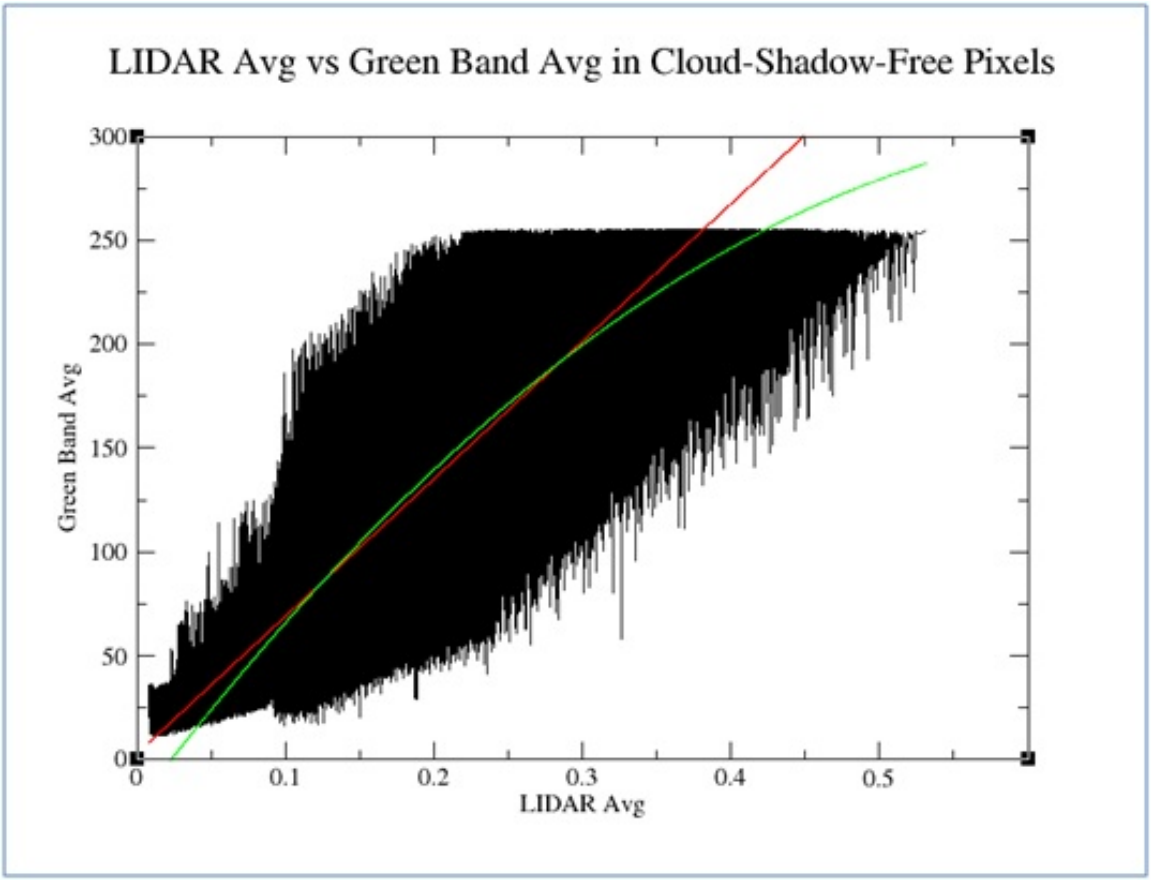

Figure 5.4: Plot of $LIDAR_{avg}$ vs. $green_{avg}$ in Cloud-Shadow-Free Pixels. The red line is the linear fit, and the green curve is the quadratic fit.

Due to the similarity of the two equations, we choose the linear fit, as it produces nearly the same visual effect, at a lower computational cost. We then use this equation to compute a new green average, which when divided by the original green average, gives us a multiplier we can use to scale the RGB triplet of the original pixel.

Figure 5.5: Image produced from using the linear transformation function.

Figure 5.6: Image produced from using the quadratic transformation function.

5.5 Color Blending between Cloud Shadow and Sunlit Pixels

As a final step in the mosaic process, we would like to attempt to make the remaining seams between adjacent flight lines and between cloud shadow and no cloud shadow regions less noticeable. Even after each of our attempts at color matching, unfortunately we cannot make a perfect transition between flight lines. However, we still need to try to fade the seams of each transition between corrected and non-corrected regions, to make them less noticeable. In order to do this, we first start by building a couple of distance maps [5], one for cloud shadow and one for no cloud shadow, for the entire output mosaic. For each pixel of the output, we would like to know the distance from this pixel to the nearest cloud shadow-covered pixel and the nearest no cloud shadow-covered pixel.

We start by initializing a 6-D array, where elements 1 and 2 are the distances to the nearest cloud and not cloud pixels, respectively, and the remaining 4 elements hold the index (location) of the nearest pixel of each type. This array is built using the cloud/not-cloud map created from the classification model to identify whether a pixel is in cloud or not in cloud, and then we proceed to fill in the distances to pixels of the opposite type. Using this information, we can then know where the cloud-to-not-cloud transition points are, which represent the area we would like to fade together.

In the case of a cloud shadow covered pixel, we would like to know the nearest unobscured pixel, and if this distance is within some predefined range, we would then like to alter the current pixel by an amount proportional to its distance to the transition. However, we do not simply want to modify the intensity of the current pixel, but rather we would like to make the current pixel more closely match the color of the material on the non-cloud shadow side. In this way we are blending the colors in the transition areas, as well as the intensity which helps to reduce the visible, quick transition between areas.

This works fairly well in areas such as within the trees, in the neighborhoods, and other areas with large variations in pixel distributions (Figures 5.7 5.8), especially cloud/not-cloud

transition areas that are within the same flight line (Figures 5.10 5.10. However, it does not work very well in relatively flat, solid colored areas such as those in Figures 5.11 and 5.12. The close-up of this roof-top shows that the sudden intensity and color variation in the two flight lines are hard to hide via blending. They are simply too far apart to trick the eye into believing they are seamless.

Figure 5.7: Close-up of cloud/not-cloud shadow transition area. Compare to Figure 5.8.

Figure 5.8: Close-up of area in Figure 5.7 after blending.

Figure 5.9: Closeup of cloud/not-cloud shadow transition area. Compare to Figure 5.10.

Figure 5.10: Close-up of area in Figure 5.9 after blending.

Figure 5.11: Closeup of cloud/not-cloud shadow transition area. Compare to Figure 5.12.

Figure 5.12: Close-up of area in Figure 5.11 after blending.

5.6 Mosaic on the Easterly Flight Lines

Until now we have been producing the image mosaic from the total set of images taken during the survey mission. It includes flight lines in all directions, which for the sake of discussion, we have been referring to as east and west flight lines. It is interesting to note that in the case our current set of data, the easterly flight lines contain less cloud and less sun effect than do the west flight lines. This is because as the plane flies towards the sun, more light is reflected back into camera lens than when the flight is away from the sun. Essentially we are getting to separate distributions of RGB (and possibly LIDAR) from each flight direction which we are using when we construct the color transformation function. This can be seen in the plot in Figure 5.13.

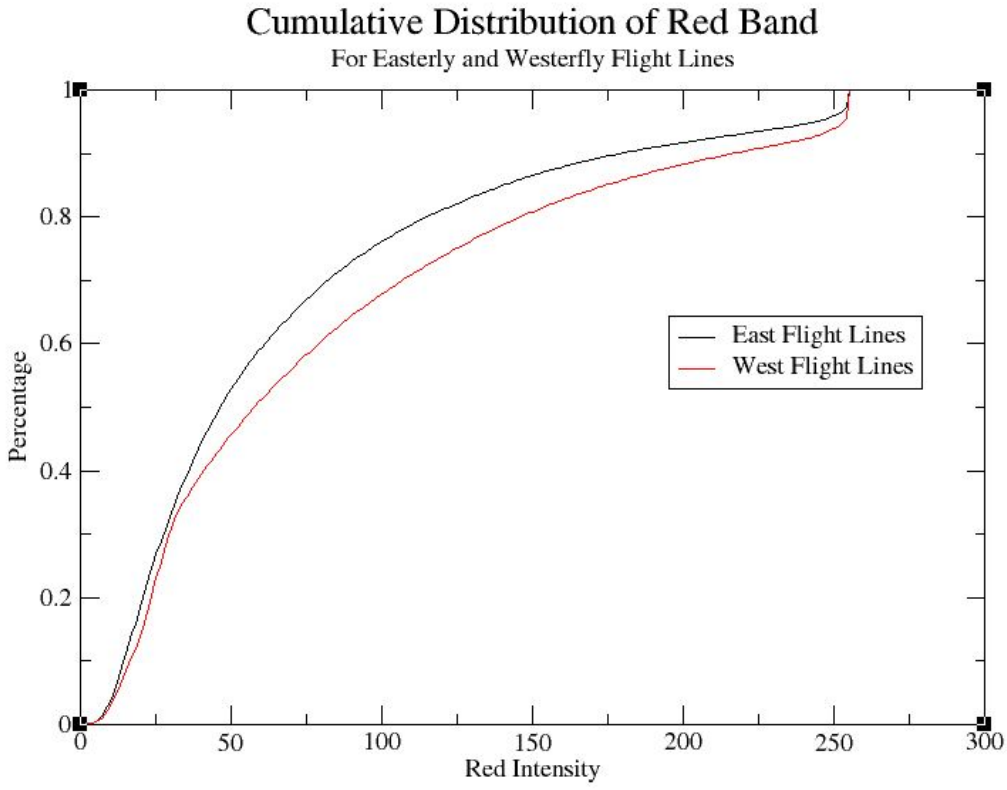

Figure 5.13: Plot of the cumulative distribution of red in both the easterly and westerly flight lines.

Instead, we would like the transformation function to be built solely on easterly flight lines, as they exhibit less reflectivity distortions. In addition to this, we would also like to build the mosaic using only the easterly flight lines, and then apply this transformation function. We are able to successfully build this mosaic because the set of data we have is complete in both the east and west directions. This may not necessarily always be the case, which is why we have been building the mosaic with the complete set of input images. However, we want to show what, if any, effect there is on the amount of sunlight reflection and it's relationship to the LIDAR. It is possible that the LIDAR sensor measures some additional intensity as the sun also radiates the same 1064 nm signal. Therefore, using only the flight lines in the direction away from the sun allows us to build the model with a single, similar distribution.

We rebuilt the models, cloud/no cloud maps, and mosaics using only easterly flight lines, with both the linear and SVM classifiers. This required retraining the models using only images from the easterly flight line, which meant generating new image masks. In the case of the linear classifier, the new model had a slightly lower cross-validation accuracy at 85.1217%, but this was practically no difference in the actual cloud/not-cloud map (Figure 5.14). The SVM classifier had a similar change, but in the opposite direction, as it increased to 89.1155%. As can be seen from the resulting mosaic in Figure 5.15, the color correction is much more accurate, and produces a more consistent looking image using only the easterly flight lines.

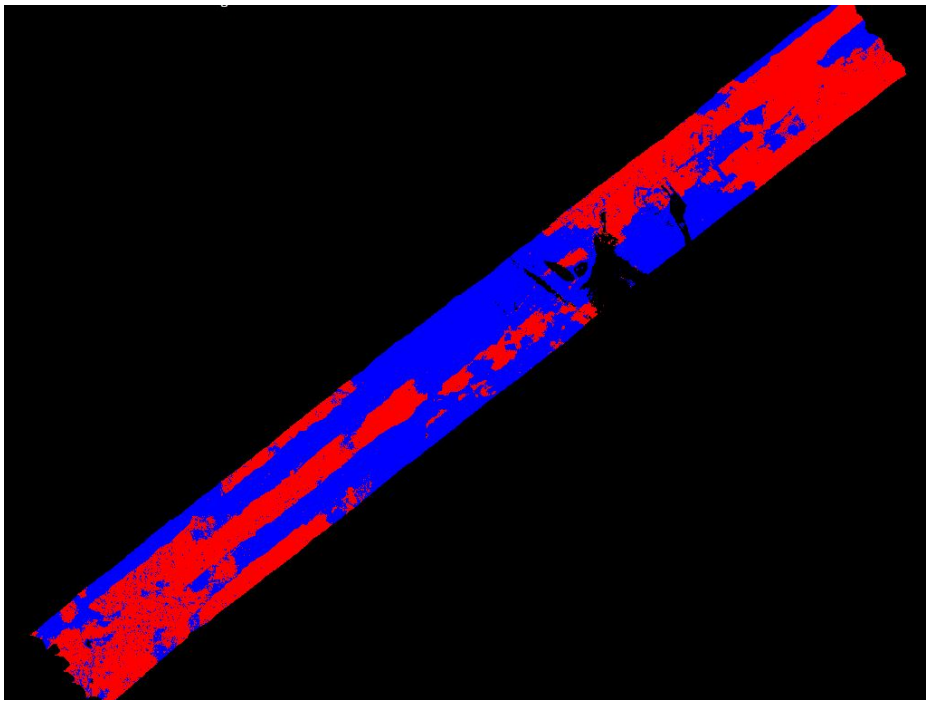

Figure 5.14: Cloud/Not-Cloud map created with the linear classifier for east flight lines only.

Figure 5.15: Mosaic with all aforementioned corrections, on only the easterly flight lines.

Figure 5.16: Close-up of the upper right quadrant of the image mosaic created using all (east and west) flight lines.

Figure 5.17: Close-up of the upper right quadrant of the image mosaic created using only easterly flight lines.

Figure 5.18: Close-up of the lower left quadrant of the image mosaic created using all (east and west) flight lines.

Figure 5.19: Close-up of the lower left quadrant of the image mosaic created using only easterly flight lines.

Chapter 6

CONCLUSION

In this research, we set out to show that the LIDAR data could be utilized in a way that would allow us to detect and remove cloud shadows from the image mosaic. We have been largely successful in accomplishing this, as can be seen in a visual comparison of the cloud/not-cloud maps and the original, problematic mosaics. We have also shown that, in order to perform many useful color corrections to this cloud shadow-covered data, it is important to know the predicted class of an output pixel. This allows us to build transformations from the two different classes of data in order to restore the image to what it would have looked liked had all of the pixels been under pure, unobscured sunlight.

The remaining artifacts in the image mosaic are due to a number of fundamental problems with correcting the color in the cloud shadow-covered areas. One thing we noticed is that the clouds seem to affect different wavelengths (colors) of light in different ways. For instance, in areas with dense trees we see that the color distribution is not the same as in the nearby parts of the same forest which are under sunlit pixels. These corrected areas appear to consist only of shades of green, with the dark yellows and oranges apparently gone. These colors are clearly seen in the same group of trees in nearby sunlit pixels. Also noticeable is the apparent overall intensity of this area. While the intensity is what we are aiming to correct with the LIDAR data, we do not have enough data to reconstruct the color distribution. Knowing how the clouds, atmosphere, and surface material affect the reflection and appearance of the different wavelengths of sunlight may be necessary to generate a more accurate model.

Also, in the beginning of this paper, we mentioned that we did not have the location of the clouds, and therefore could not build a model which was based on the geometric

relationship of the cloud, sun, and cloud shadow. Nevertheless, a model which did take into account the exact position of the sun with respect to the line of flight of the aircraft would allow us to possibly correct for the different flight paths. The easterly flight line mosaics from section x are an example of this, albeit a manual example. If we could determine this and correct for this in any direction of flight, automatically, this would be very helpful.

Yet another future research effort could reveal whether doing ground cover classification could also assist in the color transfer function. In fact, the hyper-spectral data collected on the flight is routinely used for such purposes. This would then give us the ability to, for instance, map areas of dense trees differently that we possibly would for an area of beach. This might also be very useful for the color matching algorithms, as a ground cover classification may give us a metric to determine whether an area of cloud shadow and sunlight are of the same general content. Using color transfer, we can get produce an image which more closely matches the color distribution in the trees, for instance, by using those yellows and oranges in the nearby sunlit pixels to re-color the cloud shadowed pixels.

We have also discussed a few color transformation methods, including our preferred method of fitting a function to the sunlight pixel data. However, the color transformation method presented can still be improved upon. In fact, the entire process can be improved upon if we simply had better LIDAR data. As mentioned previously, the LIDAR data is extremely noisy, as its primary design purpose was simply for building digital elevation maps.

The CZMIL project also has plans in the not-to-distant future to upgrade the entire data collection system, including the LIDAR, RGB, and hyper-spectral cameras on board. Careful consideration is being observed in the design of this new LIDAR system, with the goal of producing a LIDAR reflectance map which is higher resolution, less noisy, and better differentiates the LIDAR return signal from that of the sun. As the LIDAR is the second most important piece of data (besides the actual RGB pixel itself), the mosaic process is certain to benefit from this increased accuracy.

BIBLIOGRAPHY

[1] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm (Accessed 2/11/2011).

[2] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3 edition, September 2009.

[3] I. J. Cox, S. Roy, and S. L. Hingorani. Dynamic histogram warping of image pairs for constant image brightness. In *Proceedings of the 1995 International Conference on Image Processing (Vol.2)-Volume 2 - Volume 2*, ICIP '95, pages 366 – 369, Washington, DC, USA, 1995. IEEE Computer Society.

[4] Paul Dare. Shadow analysis in high-resolution satellite imagery of urban areas. In *Photogrammetric Engineering and Remote Sensing*, volume 71, pages 169–177. American Society for Photogrammetry and Remote Sensing, 2005.

[5] R. Fabbri, L. da F. Costa, J. C. Torelli, and O. M. Bruno. 2d euclidean distance transform algorithms: A comparative survey. *ACM Computing Surveys*, 40(1):2:1–2:44, 2008.

[6] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[7] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 2001.

[8] Jiawei Han. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[9] E.H. Helmer and B. Ruefenacht. Cloud-free satellite image mosaics with regression trees and histogram matching. *Photogrammetric Engineerin and Remote Sensing*, 71:1079–1089, 2005.

[10] Nguyen Thanh Hoan and Ryutaro Tateishi. Cloud removal of optical image using sar data for alos applications. experimenting on simulated alos data. pages 73–74, 2008.

[11] Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S. Sathiya Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear svm. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 408–415, New York, NY, USA, 2008. ACM.

[12] C. W. Hsu, C. C. Chang, and C. J. Lin. A practical guide to support vector classification. Technical report, Taipei, 2003.

[13] Yao-Hsien Huang and Chung-Hsin Liu. A practical color transfer algorithm for image sequences. *Intelligent Information Hiding and Multimedia Signal Processing, International Conference on*, 1:577–580, 2007.

[14] Wenjing Jia, Huaifeng Zhang, Xiangjian He, and Qiang Wu. A comparison on histogram based image matching methods. *Advanced Video and Signal Based Surveillance, IEEE Conference on*, 0:97, 2006.

[15] S. Kagarlitsky, Y. Moses, and Y. Hel-Or. Piecewise-consistent color mappings of images acquired under various conditions. In *Proceedings of the 12th IEEE International Conference on Computer Vision*, pages 2311–2318, 2009.

[16] S. Sathiya Keerthi, Kai-Wei Chang, and et al. A sequential dual method for large scale multi-class linear svms, 2008.

[17] Chih-Jen Lin, Ruby C. Weng, and Sathiya S. Keerthi. Trust region Newton methods for large-scale logistic regression. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 561–568, New York, NY, USA, 2007. ACM.

[18] Chung ming Wang and Yao hsien Huang. A novel color transfer algorithm for image sequences. *Journal of Information Science and Engineering*, 20:1039–1056, 2004.

[19] Hui-Fuang Ng and Yen-Wei Chu. Illumination invariant color model for image matching and object recognition. In *Proceedings of the 2008 Eighth International Conference on Intelligent Systems Design and Applications - Volume 01*, pages 95–99, Washington, DC, USA, 2008. IEEE Computer Society.

[20] Francois Pitie, Anil C. Kokaram, and Rozenn Dahyot. N-dimensional probablility density function transfer and its application to colour transfer. In *Proceedings of the Tenth IEEE International Conference on Computer Vision - Volume 2*, ICCV '05, pages 1434–1439, Washington, DC, USA, 2005. IEEE Computer Society.

[21] Erik Reinhard, Michael Ashikhmin, Bruce Gooch, and Peter Shirley. Color transfer between images. *IEEE Comput. Graph. Appl.*, 21:34–41, September 2001.

[22] J.P. Rolland, V. Vo, B. Bloss, and C.K. Abbey. Fast algorithms for histogram matching: Application to texture synthesis. *Journal of Electronic Imaging*, 9:39–45, 2001.

[23] Thiago Statella and E.A. da Silva. Shadows and clouds detection in high resolution images using mathematical morphology. In *Proceedings, ASPRS 2008 Annual Conference*, Pecora, 2008. American Society fo Photogrammetry and Remote Sensing.

[24] Wataru Takeuchi and Yoshifumi Yasuoka. Development of cloud and shadow free compositing technique with modis qkm. In *Proceedings, ASPRS 2006 Annual Conference*, Reno, 2006. American Society fo Photogrammetry and Remote Sensing.

[25] Bin Wang, Atsuo Ono, Kanako MURAMATSU, and Noboru FUJIWARA. Automated detection and removal of clouds and their shadows from landsat tm images. 1999.

[26] Chuan-Kai Yang and Li-Kai Peng. Automatic mood-transferring between color images. *IEEE Computer Graphics and Applications*, 28:52–61, 2008.

[27] Jie Ying and Liang Ji. Pattern recognition based color transfer. In *Proceedings of the International Conference on Computer Graphics, Imaging and Visualization*, pages 55–60, Washington, DC, USA, 2005. IEEE Computer Society.

[28] Guo-Xun Yuan, Kai-Wei Chang, Cho-Jui Hsieh, and Chih-Jen Lin. A Comparison of Optimization Methods and Software for Large-scale L1-regularized Linear Classification. *Journal of Machine Learning Research*, pages 3183–3234.

[29] Li Zheng, Jianqing Zhang, and Yuejun Luo. Color matching in colour remote sensing image. In *Proceedings of the First International Multi-Symposiums on Computer and Computational Sciences - Volume 1 (IMSCCS'06) - Volume 01*, IMSCCS '06, pages 303–306, Washington, DC, USA, 2006. IEEE Computer Society.